UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Geociências e Ciências Exatas - IGCE
Curso de Bacharelado em Ciências da Computação

MAICON DALL'AGNOL

TRABALHO DE REGRESSÃO

Professora: Dra. Adriane Beatriz de Souza Serapião

Rio Claro - SP

2019

1 Introdução

Este trabalho consiste em aplicar o conhecimento em regressão (predição) adquirido na disciplina Tópicos: Aprendizado de Máquina, tendo assim como objetivo:

- Escolher dois conjuntos de dados para trabalhar o problema de regressão. Separe cada *dataset* em conjunto de treinamento e conjunto de teste. Explique o seu critério de separação e o método utilizado.
- Você deverá implementar soluções para cada *dataset* usando:
  - regressão linear (ou regressão múltipla)
  - regressão polinomial
  - SVR (kernels linear, sigmoide, RBF e polinomial)
  - rede neural (MLP ou RBF).
- Descrever os parâmetros/arquiteturas de cada modelo.
- Compare os resultados (para treinamento e teste) com as medidas de desempenho SEQ, EQM, REQM, EAM e r^2, e verifique qual a melhor opção dentre os métodos implementados que melhor se ajusta a seus dados.
- Você deverá fazer a visualização dos dados originais com os dados ajustados em cada experimento, tanto para o conjunto de treinamento quanto para o de teste. Os gráficos devem conter títulos nos eixos e legenda. Comente os resultados encontrados na visualização.

2 Desenvolvimento

Para o desenvolvimento das atividades inicialmente foram escolhidos duas bases bases de dados. A primeira base a ser utilizada corresponde a dados de uma central elétrica de ciclo combinado ao longo de 6 anos tendo informações como temperatura, pressão ambiente, umidade, vácuo de exaustão e saída de energia elétrica horária líquida, sendo este ultimo o escolhido para ser predito; A segunda base é composta do histórico de tempo de 2006 a 2016 em Szeged, Hungria, contendo hora, temperatura, umidade, entre outros atributos, sendo a temperatura escolhida como atributo a ser predito.

2.1 Pré-processamento e Visualização

Para ambos *datasets* utilizou-se de um biblioteca do Python chamada Pandas Profiling que descreve e faz uma pré análise dos *datasets*, como linhas duplicadas, dados faltantes, grande variância entre os dados.

A partir desta análise notou-se que no dataset do tempo há duas variáveis com alta correlação, portanto retirou-se uma delas. Também na segunda base, há diversos atributos categóricos que foram removidos a fim de agilizar o treinamento dos algoritmos (também diminuiu-se o dataset de mais 94000 para 10000), visto que o tempo de treinamento dos algoritmos é longo.

Em ambas visualizações dos *datasets* é possivel notar uma certa linearidade entre os atributos PE e AT, para o primeiro dataset, e Humidity e Temperature, no segundo, deste modo, escolheu-se esses atributos para serem usados na visualização dos itens preditos e reais.

Para aplicação dos algoritmos de regressão os dados foram escalonados utilizando o StandardScaler.

Para separação de treino e teste utilizou-se uma divisão de 20% para teste e 80% para treino, de forma aleatória.

2.2 Regressão

Na aplicação dos algoritmos de regressão foram utilizados os algoritmos de regressão linear, regressão polinomial, SVR, utilizando os kernels linear, sigmoide, RBF e polinomial, rede neural MLP. Alguns algoritmos tiveram seus parâmetros *default* alterados para outros valores, os demais permanecem inalterados.

2.3 Avaliação

Os resultados foram medidos para os dados de treino e de teste de ambas bases. Para a primeira base a Tabela 1 corresponde aos resultados da base de teste e a Tabela 2, aos dados de treino.

Algoritmo	EQM	R^2	REQM	SEQ
Regressão Linear - Teste	0,068	0,932	0,261	130,01
SVR - RBF - Teste	0,054	0,946	0,231	102,41
SVR - Linear - Teste	0,068	0,932	0,261	130,454
SVR - Sigmoide - Teste	298,19	-0,027	17,268	570735,341
SVR - Polinomial - Teste	0,212	0,787	0,461	406,592
MLP - Teste	0,053	0,946	0,231	102,359

Tabela 1 – Avaliação dos dados de teste da central elétrica.

Algoritmo	EQM	R^2	REQM	SEQ
Regressão Linear - Treino	0,072	0,928	0,269	552,272
SVR - RBF - Treino	0,054	0,946	0,233	415,487
SVR - Linear - Treino	0,073	0,927	0,269	555,226
SVR - Sigmoide - Treino	299,318	-0,027	17,301	2290976,223
SVR - Polinomial - Treino	0,219	0,781	0,468	1675,504
MLP - Treino	0,054	0,946	0,233	416,689

Tabela 2 – Avaliação dos dados de treino da central elétrica.

Comparando as medidas para todos os algoritmos aplicados ao dataset da central elétrica é possível observar que todos, exceto o SVR - Sigmoide, apresentam bons resultados em ambas bases, ficando bem próximos aos valores reais, isto é demonstrado pelo baixo valor de EQM e alto R^2, por exemplo.

Algoritmo	EQM	R^2	REQM	SEQ
Regressão Linear - Teste	0,593	0,406	0,77	1185,398
SVR - RBF - Teste	0,6	0,399	0,774	1199,215
SVR - Linear - Teste	0,601	0,397	0,775	1202,513
SVR - Sigmoide - Teste	51226,694	-51370,514	226,333	102453388,722
SVR - Polinomial - Teste	0,698	0,3	0,835	1395,272
MLP - Teste	0,583	0,415	0,764	1166,166

Tabela 3 – Avaliação dos dados de teste do tempo.

Avaliando os resultados obtidos para o dataset do tempo, presentes nas Tabelas 3 e 4, de teste e treino, respectivamente, é possivel concluir que o mesmo não ocorre aqui, apresentando dados mais esparsos, tendo todas as medidas mais elevados, contudo ainda sim apresentam baixos valores de EQM e médio R^2.

Algoritmo	EQM	R^2	REQM	SEQ
Regressão Linear - Treino	0,585	0,416	0,765	4677,922
SVR - RBF - Treino	0,577	0,423	0,76	4618,342
SVR - Linear - Treino	0,592	0,408	0,769	4736,851
SVR - Sigmoide - Treino	52816,201	-52785,446	229,818	422529610,722
SVR - Polinomial - Treino	0,755	0,245	0,869	6040,982
MLP - Treino	0,575	0,425	0,758	4599,792

Tabela 4 – Avaliação dos dados de treino do tempo.

Olhando para os resultado nas bases de teste e treino, para a primeira base de dados os algoritmos MLP e SVR - RBF apresentaram resultados bem próximos, com uma ligeira vantagem para o MLP. Na segunda base os algoritmos MLP e Regressão Linear tiveram também resultados muito próximos, novamente com uma pequena vantagem para o MLP. Portanto, em termos gerais para os dois *datasets* o MLP se saiu melhor.

Regressao1

May 29, 2019

1 0. Introdução

Trabalho:
Aluno: Maicon Dall'Agnol
R.A.: 151161868
Disciplina: Tópico em Aprendizado de Máquina
Objetivos :

- Escolha dois conjuntos de dados para trabalhar o problema de regressão. Separe cada dataset em conjunto de treinamento e conjunto de teste. Explique o seu critério de separação e o método utilizado.
- Você deverá implementar soluções para cada dataset usando:
- – regressão linear (ou regressão múltipla)
- – regressão polinomial
- – SVR (use os kernels linear, sigmoide, RBF e polinomial)
- – rede neural (MLP ou RBF).
- Descreva os parâmetros/arquiteturas de cada modelo.
- Compare os resultados (para treinamento e teste) com as medidas de desempenho SEQ, EQM, REQM, EAM e rš , e verifique qual a melhor opção dentre os métodos implementados que melhor se ajusta a seus dados.
- Você deverá fazer a visualização dos dados originais com os dados ajustados em cada experimento, tanto para o conjunto de treinamento quanto para o de teste. Os gráficos devem conter títulos nos eixos e legenda. Comente os resultados encontrados na visualização.

1.1 0.1 Dependências

Para realização da tarefa foram utilizados as seguintes bibliotecas:

```
In [1]: #Utils
        import pandas as pd
        import numpy as np
        import pandas_profiling
```

```
import math

#Preprocess
from sklearn.preprocessing import StandardScaler

# Split
from sklearn.model_selection import train_test_split

# Regressores
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.neural_network import MLPRegressor

#Metricas
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error

#Visualização
import seaborn as sns
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

2 1. Dados

O conjunto de dados contém 9568 pontos de dados coletados de uma Central Elétrica de Ciclo Combinado ao longo de 6 anos (2006-2011)

Fonte: https://archive.ics.uci.edu/ml/datasets/Combined+Cycle+Power+Plant

2.1 1.1 Informações sobre os dados:

Atributos:

- Temperatura (T) na faixa de 1,81 ř C e 37,11 ř C,
- Pressão ambiente (AP) na faixa de 992,89-1033,30 milibar,
- Umidade Relativa (UR) na faixa de 25,56% a 100,16 %
- Vácuo de exaustão (V) na faixa de 25,36-81,56 cm Hg
- Saída de energia elétrica horária líquida (EP) 420,26-495,76 MW

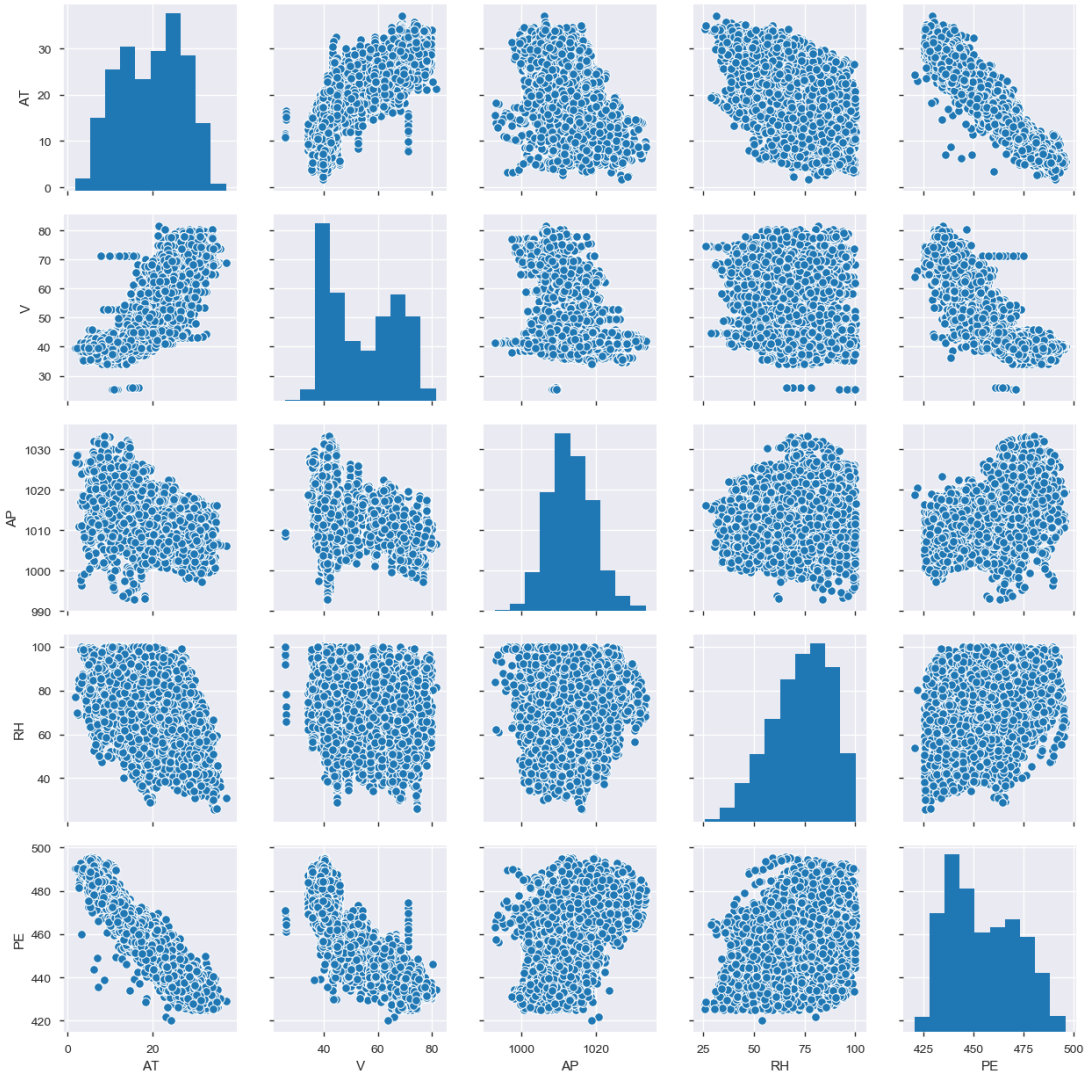
2.2 Importando Dataset

```
In [2]: data_raw = pd.read_excel('dados/Folds5x2_pp.xlsx')

In [3]: pandas_profiling.ProfileReport(data_raw)

Out[3]: <pandas_profiling.ProfileReport at 0x7f324cf2fd68>
```

2.3 Visualização

```
In [4]: sns.pairplot(data_raw)
```

```
Out[4]: <seaborn.axisgrid.PairGrid at 0x7f324a4f0ba8>
```

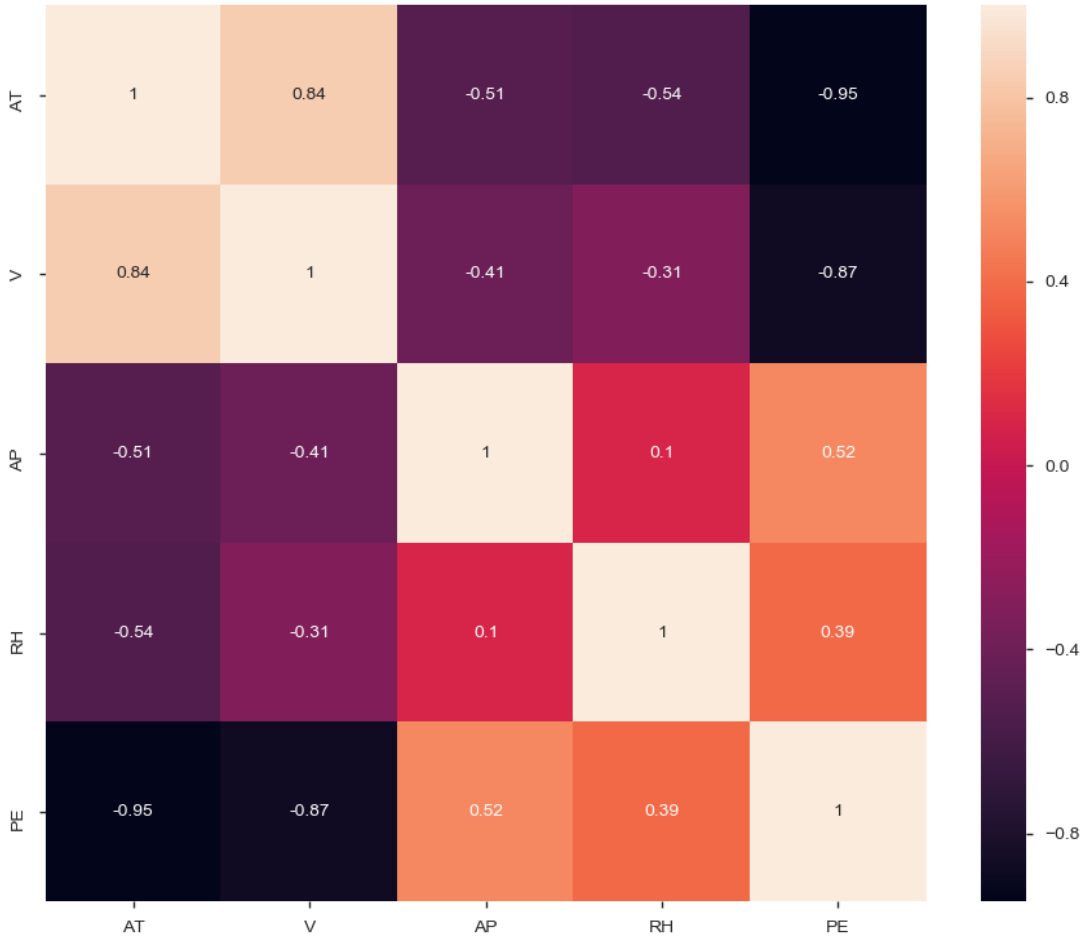



```
In [5]: plt.clf()
```

```
<Figure size 800x550 with 0 Axes>
```

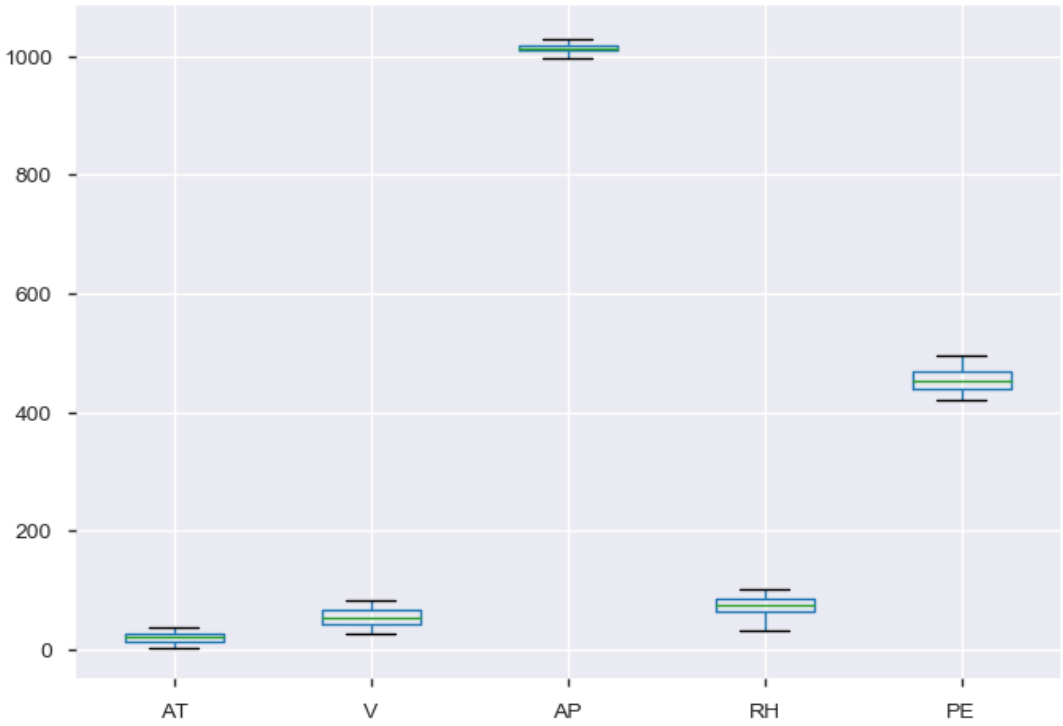
```
In [6]: plt.subplots(figsize=(11, 9))
        sns.heatmap(data_raw.corr(), annot=True)
```

```
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x7f324cd1f898>
```



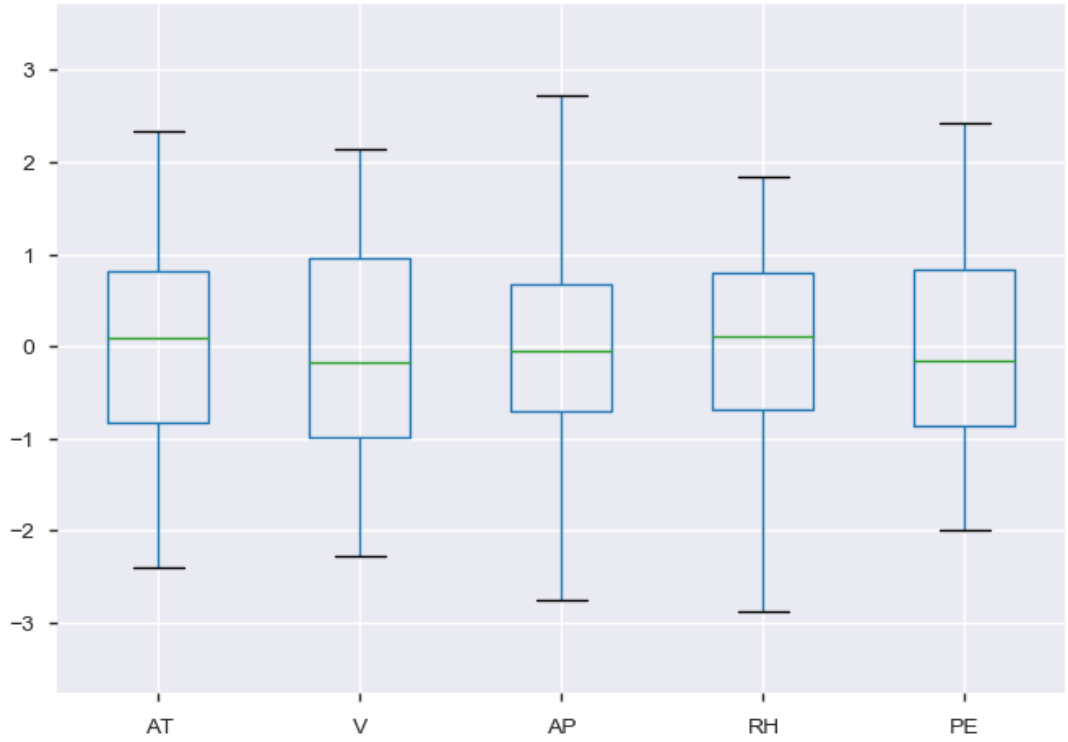
In [7]: `data_raw.plot.box()`

Out[7]: `<matplotlib.axes._subplots.AxesSubplot at 0x7f324c950128>`



2.4 Escalonando

```
In [8]: scaler = StandardScaler().fit(data_raw)
        data_scaled = scaler.transform(data_raw)

In [9]: data_scaled_df = pd.DataFrame(data_scaled, columns=['AT','V','AP','RH','PE'])

In [10]: data_scaled_df.head()

Out[10]:          AT         V        AP        RH        PE
         0 -0.629519 -0.987297  1.820488 -0.009519  0.521208
         1  0.741909  0.681045  1.141863 -0.974621 -0.585664
         2 -1.951297 -1.173018 -0.185078  1.289840  2.003679
         3  0.162205  0.237203 -0.508393  0.228160 -0.462028
         4 -1.185069 -1.322539 -0.678470  1.596699  1.144666

In [11]: data_scaled_df.plot.box()

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x7f324c8283c8>
```



2.5 Utilidades

```
In [12]: lista_metricas_treino = []
         lista_metricas_teste = []
```

```
In [13]: def metricas(y_true, y_pred, alg):
             r2 = r2_score(y_true, y_pred)
             eqm = mean_squared_error(y_true, y_pred)
             seq = len(y_true)*eqm
             reqm = math.sqrt(eqm)

             return {'Algoritmo':alg, 'R2':r2, 'EQM':eqm, 'REQM':reqm, 'SEQ':seq}
```

2.6 Separando conjuntos de Treino e Teste

Para a separação utilizou-se do train_test_split que divide o conjunto em treino e teste aleatóriamente

```
In [14]: train, test = train_test_split(data_scaled_df, test_size = 0.2, shuffle=True)

         x_train = train.drop(columns=['PE'])
         y_train = train['PE']
```

```
x_test = test.drop(columns=['PE'])
y_test = test['PE']
```

2.7 Aplicando a Regressão

2.7.1 Regressão Linear

```
In [15]: lire = LinearRegression()
```

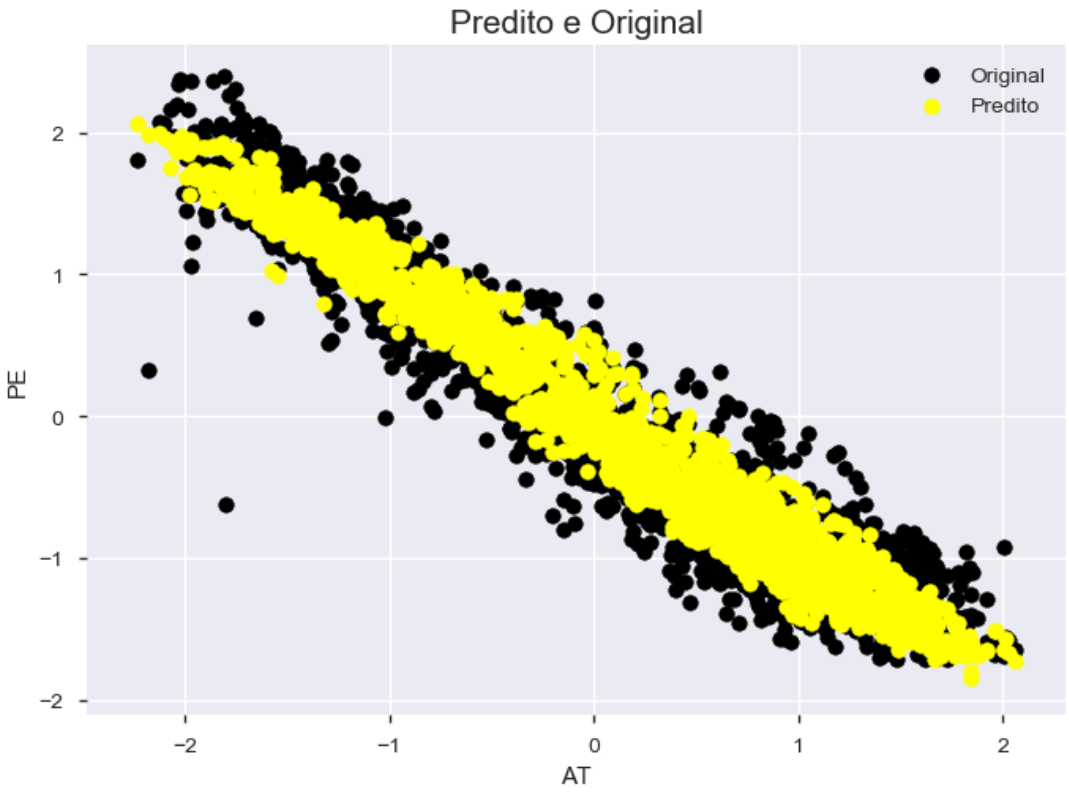
```
In [16]: lire.fit(x_train, y_train)
```

```
Out[16]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
                          normalize=False)
```

2.8 Avaliação para Teste

```
In [17]: y_pred = lire.predict(x_test)
         linear_metricas = metricas(y_test, y_pred, 'Regressão Linear - Teste')
         lista_metricas_teste.append(linear_metricas)
```

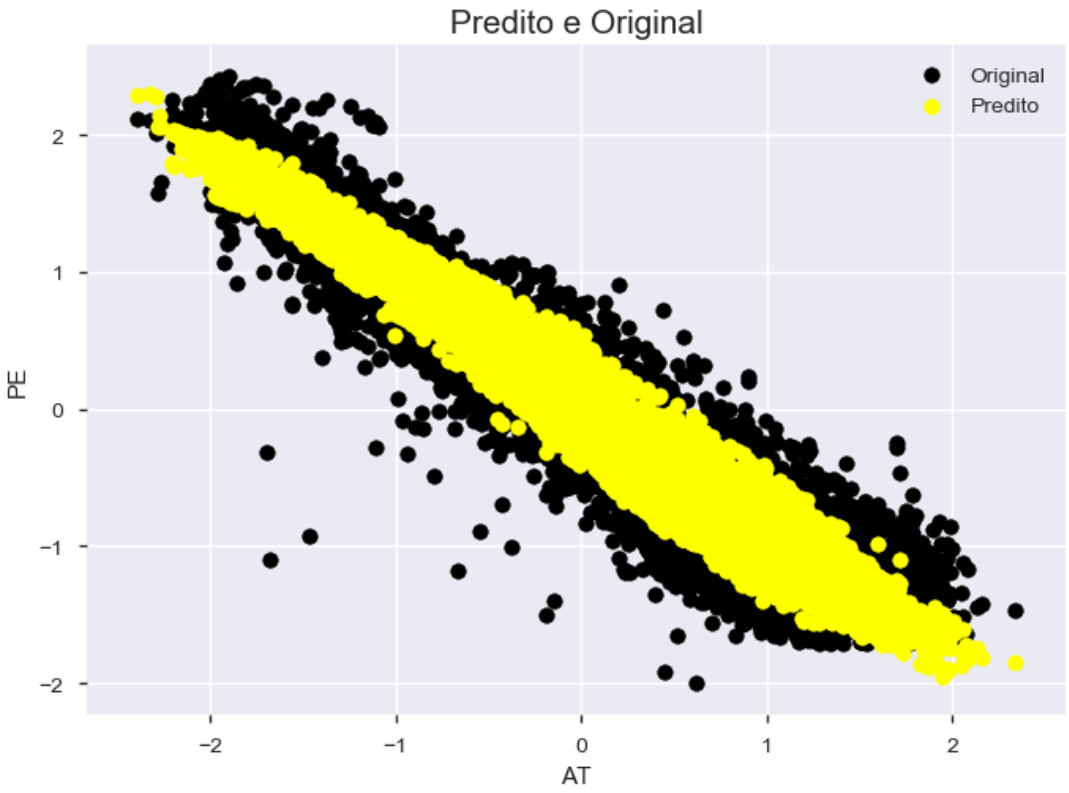
```
In [18]: plt.scatter(x_test['AT'], y_test,  color='black')
         plt.scatter(x_test['AT'], y_pred, color='yellow')
         plt.xlabel("AT")
         plt.ylabel("PE")
         plt.title('Predito e Original',fontsize=15)
         plt.legend(['Original', 'Predito'])
         plt.show()
```



2.9 Avaliação para Treino

```
In [19]: y_pred = lire.predict(x_train)
         linear_metricas = metricas(y_train, y_pred, 'Regressão Linear - Treino')
         lista_metricas_treino.append(linear_metricas)
```

```
In [20]: plt.scatter(x_train['AT'], y_train,  color='black')
         plt.scatter(x_train['AT'], y_pred, color='yellow')
         plt.xlabel("AT")
         plt.ylabel("PE")
         plt.title('Predito e Original',fontsize=15)
         plt.legend(['Original', 'Predito'])
         plt.show()
```



2.10 SVR

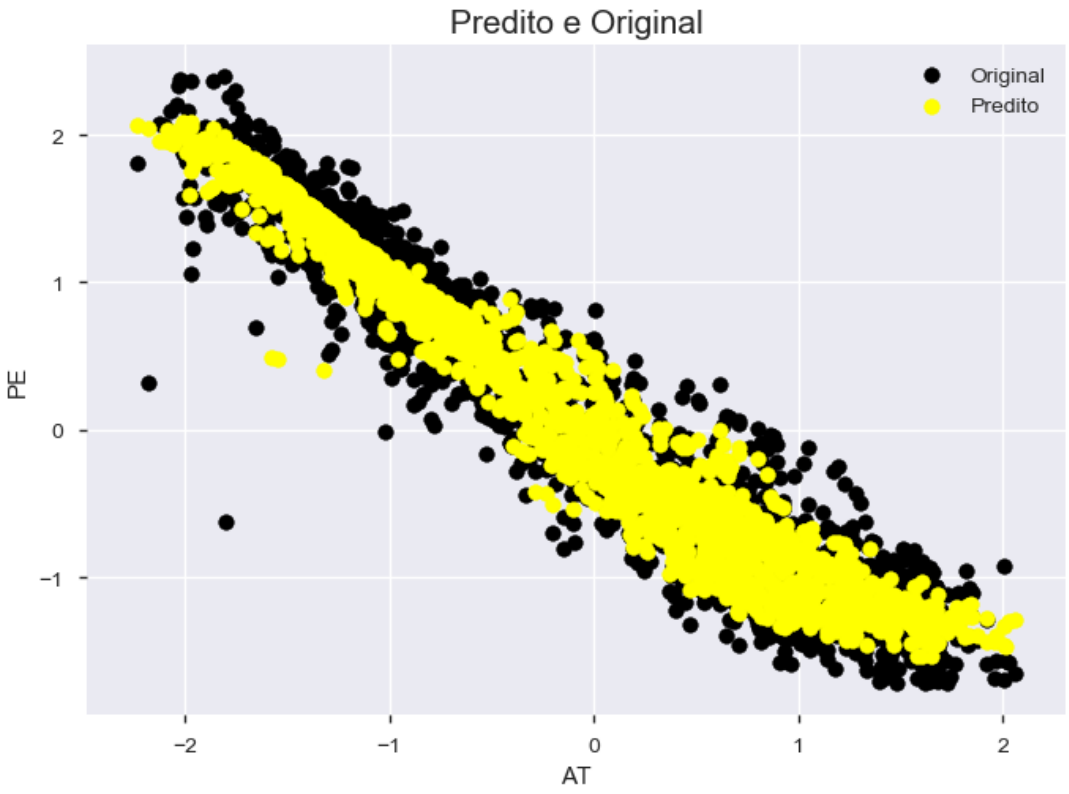
2.10.1 Kernel RBF

```
In [21]: svr_reg = SVR(kernel='rbf')

In [22]: svr_reg.fit(x_train, y_train)

Out[22]: SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
           gamma='auto_deprecated', kernel='rbf', max_iter=-1, shrinking=True,
           tol=0.001, verbose=False)
```

2.11 Avaliação para Teste

```
In [23]: y_pred = svr_reg.predict(x_test)
         svr_metricas = metricas(y_test, y_pred, 'SVR - RBF - Teste')
         lista_metricas_teste.append(svr_metricas)

In [24]: plt.scatter(x_test['AT'], y_test,  color='black')
         plt.scatter(x_test['AT'], y_pred, color='yellow')
         plt.xlabel("AT")
         plt.ylabel("PE")
```

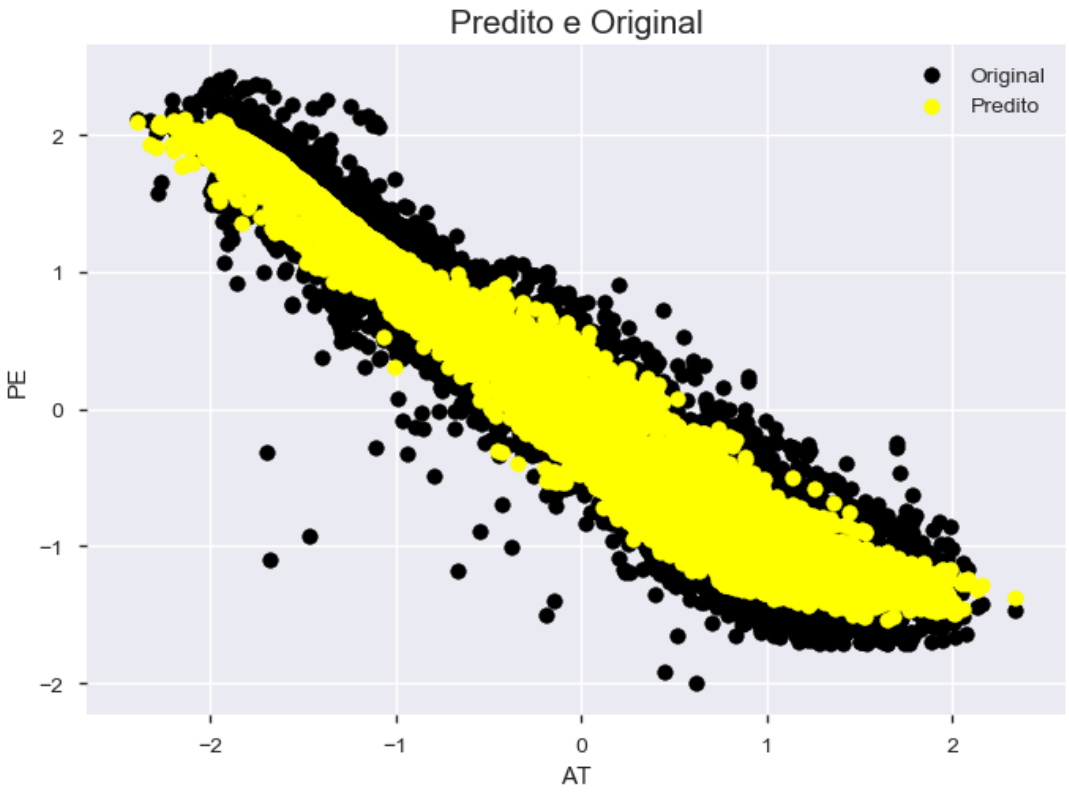
```
plt.title('Predito e Original',fontsize=15)
plt.legend(['Original', 'Predito'])
plt.show()
```



2.12 Avaliação para Treino

In [25]:
```
y_pred = svr_reg.predict(x_train)
svr_metricas = metricas(y_train, y_pred, 'SVR - RBF - Treino')
lista_metricas_treino.append(svr_metricas)
```

In [26]:
```
plt.scatter(x_train['AT'], y_train,  color='black')
plt.scatter(x_train['AT'], y_pred, color='yellow')
plt.xlabel("AT")
plt.ylabel("PE")
plt.title('Predito e Original',fontsize=15)
plt.legend(['Original', 'Predito'])
plt.show()
```



2.12.1 Kernel Linear

```
In [27]: svr_reg = SVR(kernel='linear')
```

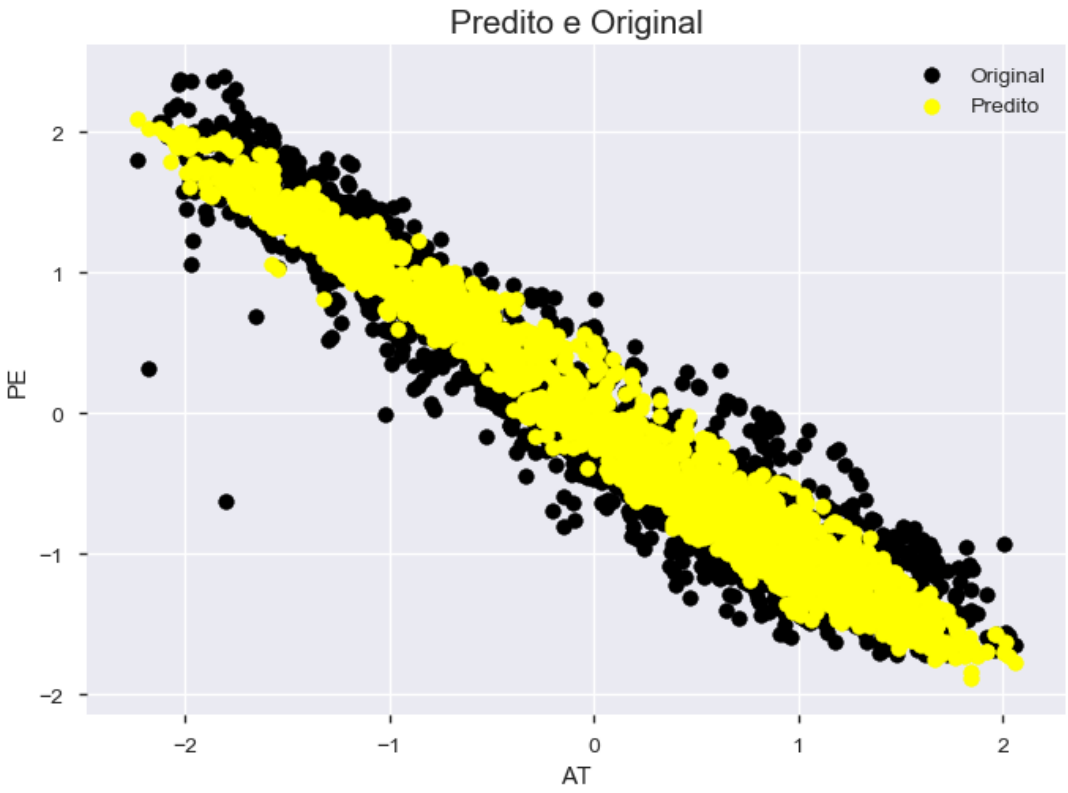
```
In [28]: svr_reg.fit(x_train, y_train)
```

```
Out[28]: SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
           gamma='auto_deprecated', kernel='linear', max_iter=-1, shrinking=True,
           tol=0.001, verbose=False)
```

2.13 Avaliação para Teste

```
In [29]: y_pred = svr_reg.predict(x_test)
         metricas_svr = metricas(y_test, y_pred, 'SVR - Linear - Teste')
         lista_metricas_teste.append(metricas_svr)
```

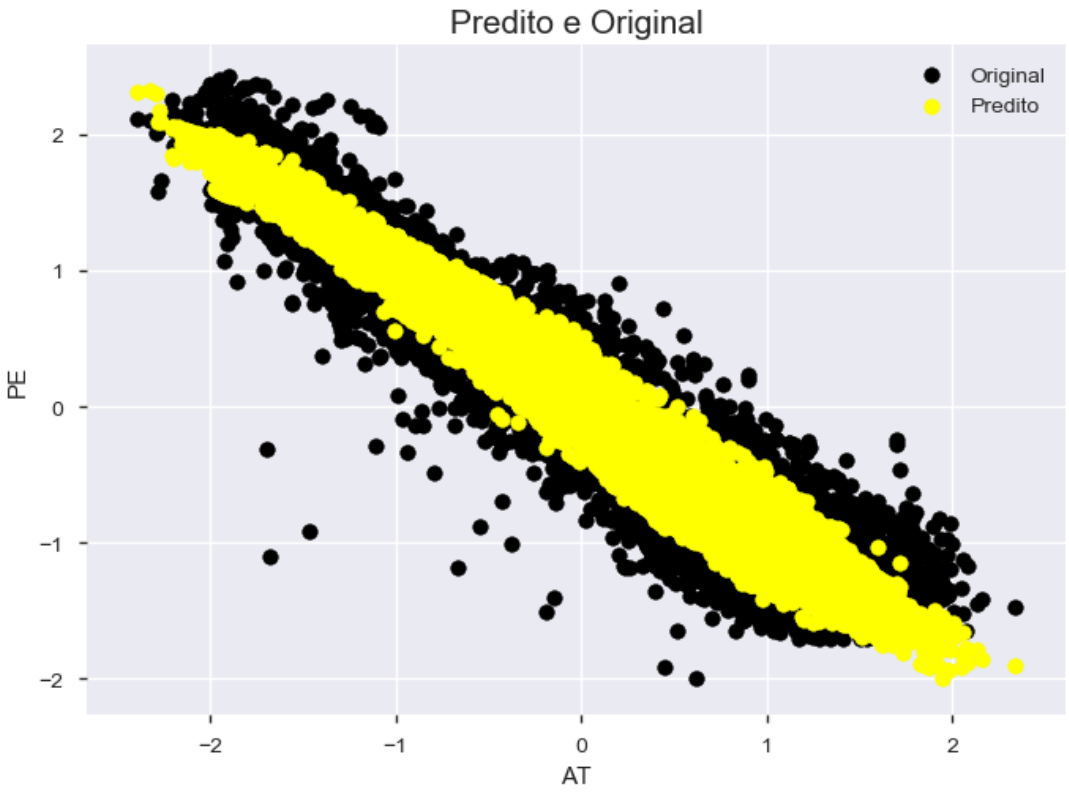
```
In [30]: plt.scatter(x_test['AT'], y_test,  color='black')
         plt.scatter(x_test['AT'], y_pred, color='yellow')
         plt.xlabel("AT")
         plt.ylabel("PE")
         plt.title('Predito e Original',fontsize=15)
         plt.legend(['Original', 'Predito'])
         plt.show()
```



2.14 Avaliação para Treino

```
In [31]: y_pred = svr_reg.predict(x_train)
         svr_metricas = metricas(y_train, y_pred, 'SVR - Linear - Treino')
         lista_metricas_treino.append(svr_metricas)
```

```
In [32]: plt.scatter(x_train['AT'], y_train,  color='black')
         plt.scatter(x_train['AT'], y_pred, color='yellow')
         plt.xlabel("AT")
         plt.ylabel("PE")
         plt.title('Predito e Original',fontsize=15)
         plt.legend(['Original', 'Predito'])
         plt.show()
```



2.14.1 Kernel Sigmoide

```
In [33]: train, test = train_test_split(data_raw, test_size = 0.2, shuffle=True)

         x_train_sig = train.drop(columns=['PE'])
         y_train_sig  = train['PE']

         x_test_sig  = test.drop(columns=['PE'])
         y_test_sig  = test['PE']

In [34]: svr_reg = SVR(kernel='sigmoid')

In [35]: svr_reg.fit(x_train_sig , y_train_sig )

Out[35]: SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
           gamma='auto_deprecated', kernel='sigmoid', max_iter=-1, shrinking=True,
           tol=0.001, verbose=False)
```

2.15 Avaliação para Teste

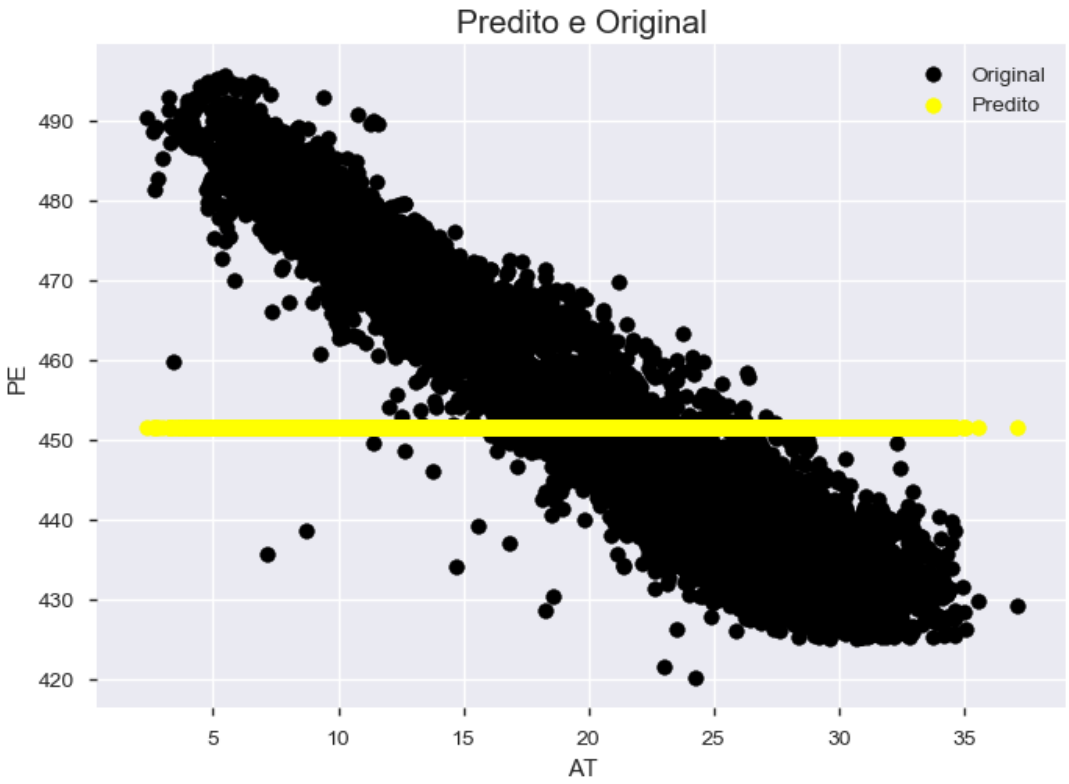
```
In [36]: y_pred_sig  = svr_reg.predict(x_test_sig)
         metricas_svr = metricas(y_test_sig , y_pred_sig , 'SVR - Sigmoide - Teste')
         lista_metricas_teste.append(metricas_svr)
```

```
In [37]: plt.scatter(x_test_sig ['AT'], y_test_sig ,  color='black')
         plt.scatter(x_test_sig ['AT'], y_pred_sig , color='yellow')
         plt.xlabel("AT")
         plt.ylabel("PE")
         plt.title('Predito e Original',fontsize=15)
         plt.legend(['Original', 'Predito'])
         plt.show()
```

2.16 Avaliação para Treino

```
In [38]: y_pred_sig  = svr_reg.predict(x_train_sig)
         svr_metricas = metricas(y_train_sig , y_pred_sig , 'SVR - Sigmoide - Treino')
         lista_metricas_treino.append(svr_metricas)
```

```
In [39]: plt.scatter(x_train_sig ['AT'], y_train_sig ,  color='black')
         plt.scatter(x_train_sig ['AT'], y_pred_sig , color='yellow')
         plt.xlabel("AT")
         plt.ylabel("PE")
         plt.title('Predito e Original',fontsize=15)
         plt.legend(['Original', 'Predito'])
         plt.show()
```

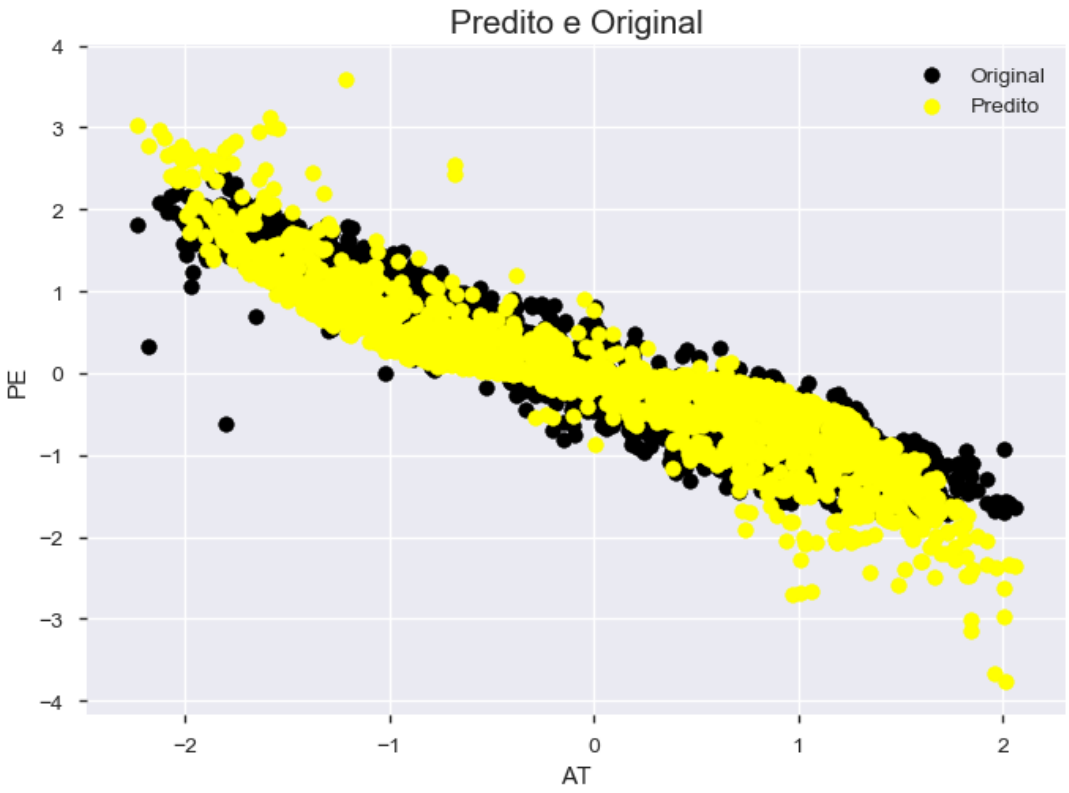


2.16.1 Kernel Polinomial

```
In [40]: svr_reg = SVR(kernel='poly', degree=3)

In [41]: svr_reg.fit(x_train, y_train)

Out[41]: SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
           gamma='auto_deprecated', kernel='poly', max_iter=-1, shrinking=True,
           tol=0.001, verbose=False)
```

2.17 Avaliação para Teste

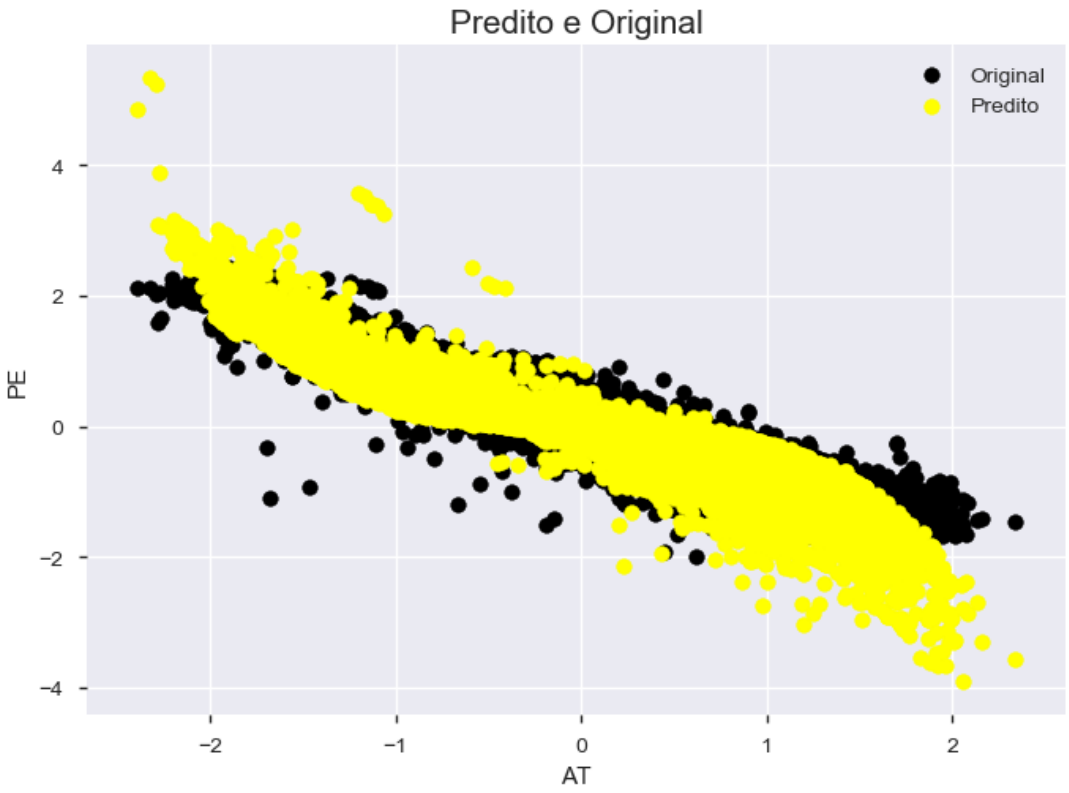
```
In [42]: y_pred = svr_reg.predict(x_test)
         svr_metricas = metricas(y_test, y_pred, 'SVR - Polinomial - Teste')
         lista_metricas_teste.append(svr_metricas)

In [43]: plt.scatter(x_test['AT'], y_test,  color='black')
         plt.scatter(x_test['AT'], y_pred, color='yellow')
         plt.xlabel("AT")
         plt.ylabel("PE")
         plt.title('Predito e Original',fontsize=15)
         plt.legend(['Original', 'Predito'])
         plt.show()
```



2.18 Avaliação para Treino

```
In [44]: y_pred = svr_reg.predict(x_train)
         svr_metricas = metricas(y_train, y_pred, 'SVR - Polinomial - Treino')
         lista_metricas_treino.append(svr_metricas)
```

```
In [45]: plt.scatter(x_train['AT'], y_train,  color='black')
         plt.scatter(x_train['AT'], y_pred, color='yellow')
         plt.xlabel("AT")
         plt.ylabel("PE")
         plt.title('Predito e Original',fontsize=15)
         plt.legend(['Original', 'Predito'])
         plt.show()
```



2.19 Redes Neurais

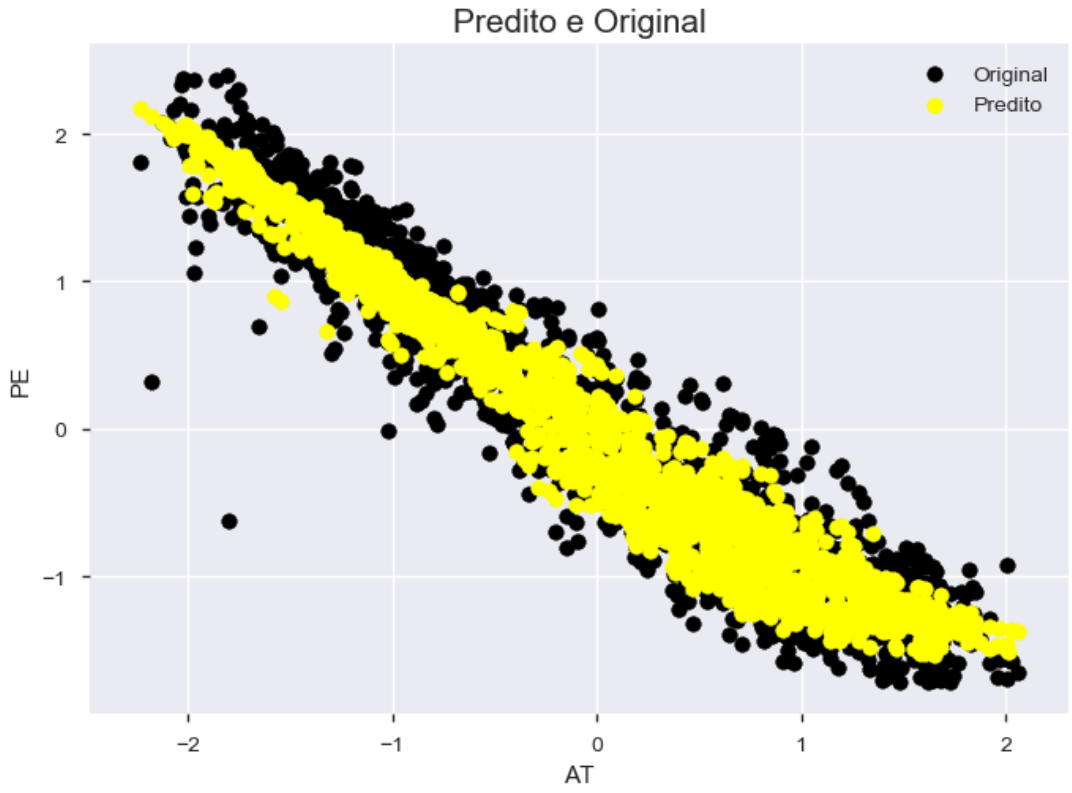
2.19.1 Kernel Linear

```
In [46]: mlp_reg = MLPRegressor()

In [47]: mlp_reg.fit(x_train, y_train)

Out[47]: MLPRegressor(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
                      beta_2=0.999, early_stopping=False, epsilon=1e-08,
                      hidden_layer_sizes=(100,), learning_rate='constant',
                      learning_rate_init=0.001, max_iter=200, momentum=0.9,
                      n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
                      random_state=None, shuffle=True, solver='adam', tol=0.0001,
                      validation_fraction=0.1, verbose=False, warm_start=False)
```

2.20 Avaliação para Teste

```
In [48]: y_pred = mlp_reg.predict(x_test)
         mlp_metricas = metricas(y_test, y_pred, 'MLP - Teste')
         lista_metricas_teste.append(mlp_metricas)
```

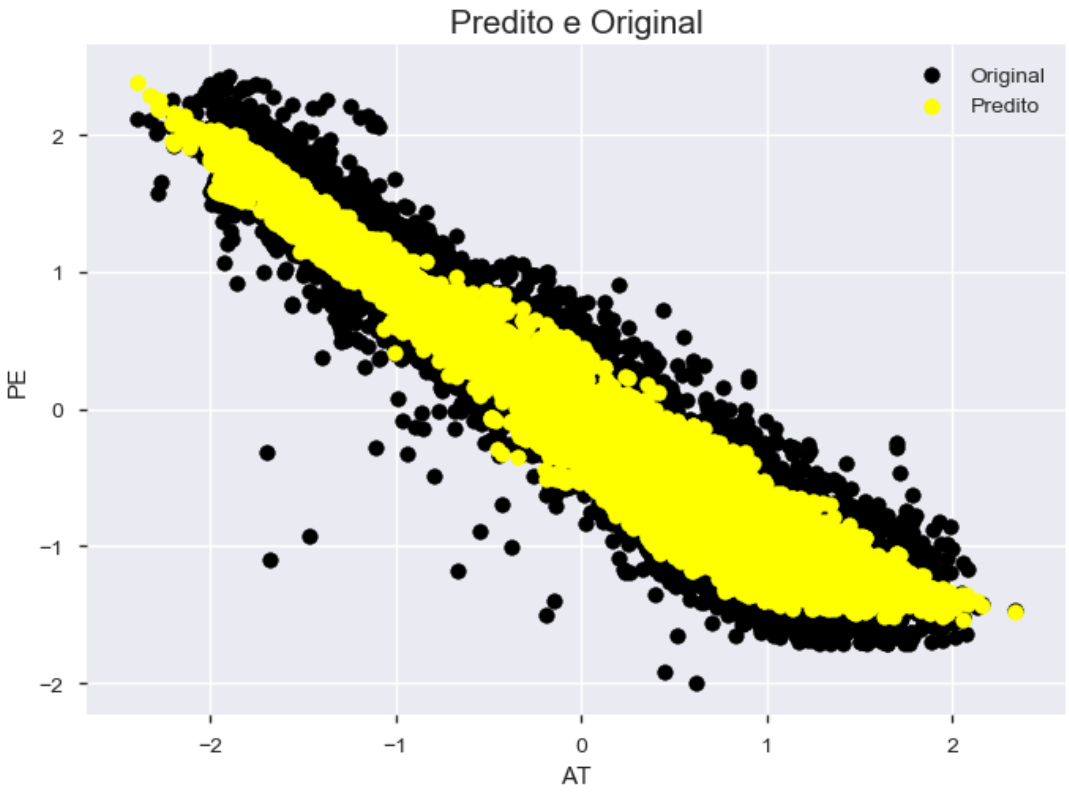
```
In [49]: plt.scatter(x_test['AT'], y_test,  color='black')
         plt.scatter(x_test['AT'], y_pred, color='yellow')
         plt.xlabel("AT")
         plt.ylabel("PE")
         plt.title('Predito e Original',fontsize=15)
         plt.legend(['Original', 'Predito'])
         plt.show()
```



2.21 Avaliação para Treino

```
In [50]: y_pred = mlp_reg.predict(x_train)
         mlp_metricas = metricas(y_train, y_pred, 'MLP - Treino')
         lista_metricas_treino.append(mlp_metricas)
```

```
In [51]: plt.scatter(x_train['AT'], y_train,  color='black')
         plt.scatter(x_train['AT'], y_pred, color='yellow')
         plt.xlabel("AT")
         plt.ylabel("PE")
         plt.title('Predito e Original',fontsize=15)
         plt.legend(['Original', 'Predito'])
         plt.show()
```



3 Resultados

```
In [90]: metricas_teste = pd.DataFrame(lista_metricas_teste)
         metricas_teste
```

Out[90]:

	Algoritmo	EQM	R2	REQM	SEQ
0	Regressão Linear - Teste	0.067926	0.931975	0.260626	130.009759
1	SVR - RBF - Teste	0.053505	0.946416	0.231313	102.409518
2	SVR - Linear - Teste	0.068158	0.931742	0.261070	130.454018
3	SVR - Sigmoide - Teste	298.189833	-0.027406	17.268174	570735.340650
4	SVR - Polinomial - Teste	0.212430	0.787258	0.460902	406.591798
5	MLP - Teste	0.053479	0.946442	0.231255	102.358966

```
In [91]: metricas_teste = round(metricas_teste, 3)
```

```
In [92]: metricas_teste
```

Out[92]:

	Algoritmo	EQM	R2	REQM	SEQ
0	Regressão Linear - Teste	0.068	0.932	0.261	130.010
1	SVR - RBF - Teste	0.054	0.946	0.231	102.410
2	SVR - Linear - Teste	0.068	0.932	0.261	130.454

```
3    SVR - Sigmoide - Teste  298.190 -0.027  17.268  570735.341
4  SVR - Polinomial - Teste    0.212  0.787   0.461     406.592
5              MLP - Teste    0.053  0.946   0.231     102.359
```

In [93]:
```
metricas_teste.to_excel('regressao1_metricas_teste.xlsx')
```

In [96]:
```
metricas_treino = pd.DataFrame(lista_metricas_treino)
metricas_treino
```

Out[96]:

	Algoritmo	EQM	R2	REQM	SEQ
0	Regressão Linear - Treino	0.072155	0.927856	0.268616	5.522719e+02
1	SVR - RBF - Treino	0.054284	0.945725	0.232988	4.154867e+02
2	SVR - Linear - Treino	0.072541	0.927470	0.269334	5.552260e+02
3	SVR - Sigmoide - Treino	299.317510	-0.026797	17.300795	2.290976e+06
4	SVR - Polinomial - Treino	0.218906	0.781128	0.467874	1.675504e+03
5	MLP - Treino	0.054441	0.945568	0.233325	4.166893e+02

In [97]:
```
metricas_treino = round(metricas_treino, 3)
```

In [98]:
```
metricas_treino.to_excel('regressao1_metricas_treino.xlsx')
```

Regressao2

May 29, 2019

1 0. Introdução

Trabalho:
Aluno: Maicon Dall'Agnol
R.A.: 151161868
Disciplina: Tópico em Aprendizado de Máquina
Objetivos :

- Escolha dois conjuntos de dados para trabalhar o problema de regressão. Separe cada dataset em conjunto de treinamento e conjunto de teste. Explique o seu critério de separação e o método utilizado.
- Você deverá implementar soluções para cada dataset usando:
- – regressão linear (ou regressão múltipla)
- – regressão polinomial
- – SVR (use os kernels linear, sigmoide, RBF e polinomial)
- – rede neural (MLP ou RBF).
- Descreva os parâmetros/arquiteturas de cada modelo.
- Compare os resultados (para treinamento e teste) com as medidas de desempenho SEQ, EQM, REQM, EAM e rš, e verifique qual a melhor opção dentre os métodos implementados que melhor se ajusta a seus dados.
- Você deverá fazer a visualização dos dados originais com os dados ajustados em cada experimento, tanto para o conjunto de treinamento quanto para o de teste. Os gráficos devem conter títulos nos eixos e legenda. Comente os resultados encontrados na visualização.

1.1 0.1 Dependências

Para realização da tarefa foram utilizados as seguintes bibliotecas:

```
In [1]: #Utils
        import pandas as pd
        import numpy as np
        import pandas_profiling
```

```
import math

#Preprocess
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder

# Split
from sklearn.model_selection import train_test_split

# Regressores
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.neural_network import MLPRegressor

#Metricas
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error

#Visualização
import seaborn as sns
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

2 1. Dados

Histórico de tempo em Szeged, Hungria - de 2006 a 2016
Fonte: https://www.kaggle.com/budincsevity/szeged-weather

2.1 1.1 Informações sobre os dados:

Atributos:

- time
- summary
- precipType
- temperature
- apparentTemperature
- humidity
- windSpeed
- windBearing
- visibility
- loudCover
- pressure

2.2 Importando Dataset

```
In [2]: data = pd.read_csv('dados/weatherHistory.csv')
```

```
In [3]: data = data.sample(10000).reset_index(drop=True,)
```

```
In [4]: pandas_profiling.ProfileReport(data)
```

```
Out[4]: <pandas_profiling.ProfileReport at 0x7fd9ec1eaf98>
```

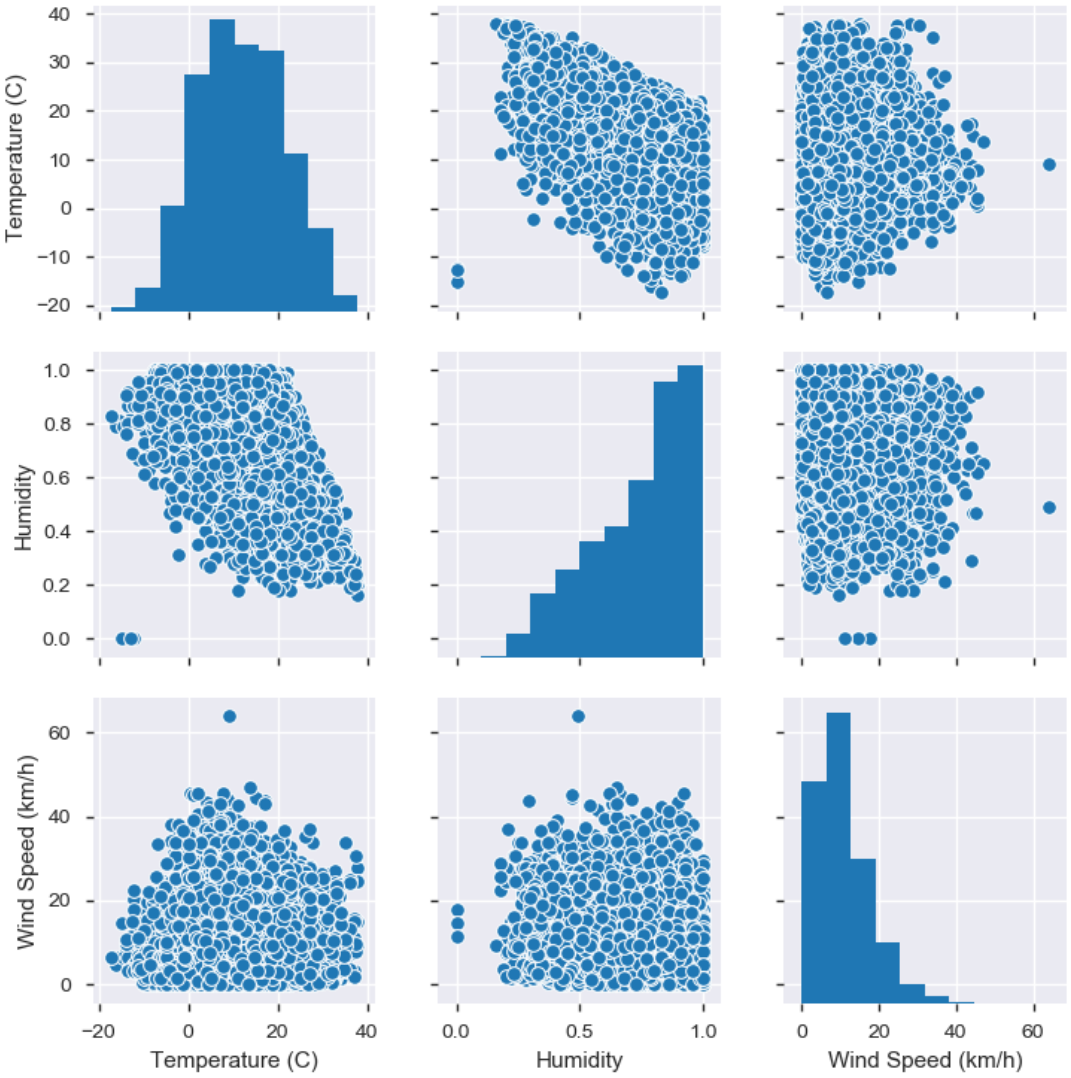
```
In [5]: data_raw = data.copy()
```

```
In [6]: data_raw.drop(columns=['Formatted Date','Precip Type','Loud Cover','Apparent Temperatu
```

2.3 Visualização

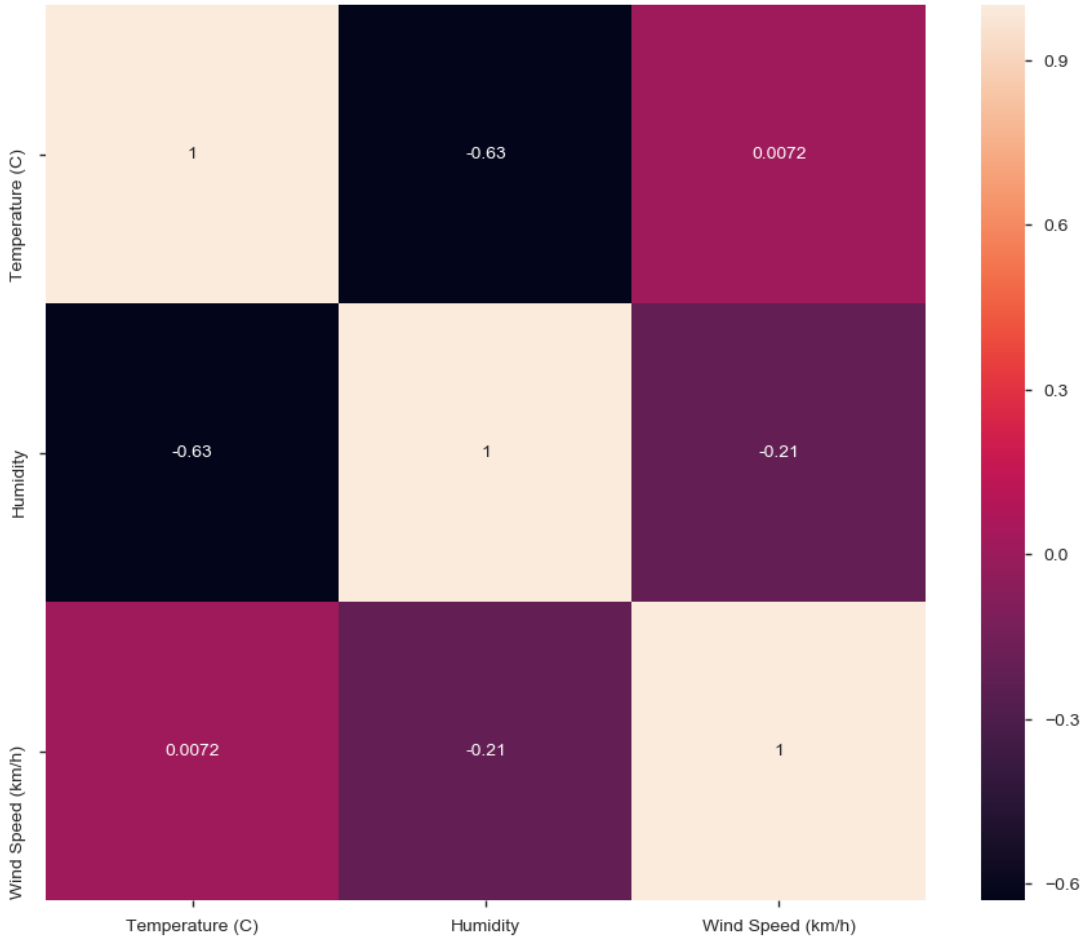
```
In [7]: sns.pairplot(data_raw)
```

```
Out[7]: <seaborn.axisgrid.PairGrid at 0x7fd9bf671c88>
```



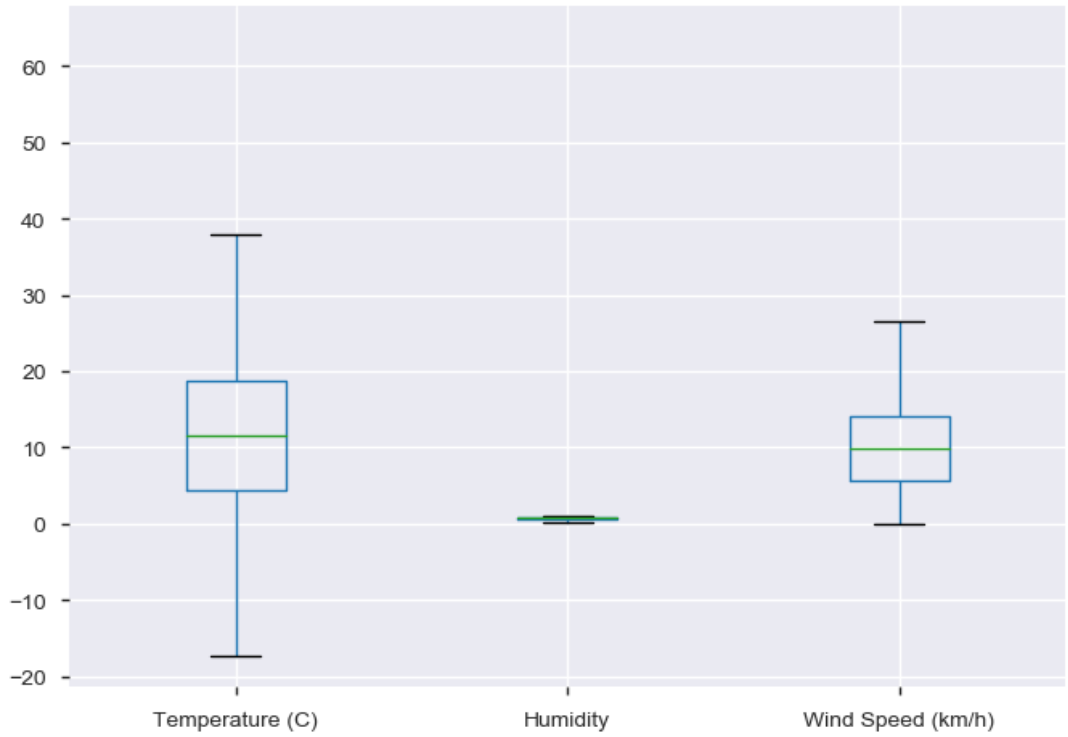
In [8]:
```
plt.subplots(figsize=(11, 9))
sns.heatmap(data_raw.corr(), annot=True)
```

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd9bf4c5860>



In [9]:
```
data_raw.plot.box()
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd9bc2ea390>



2.4 Escalonando

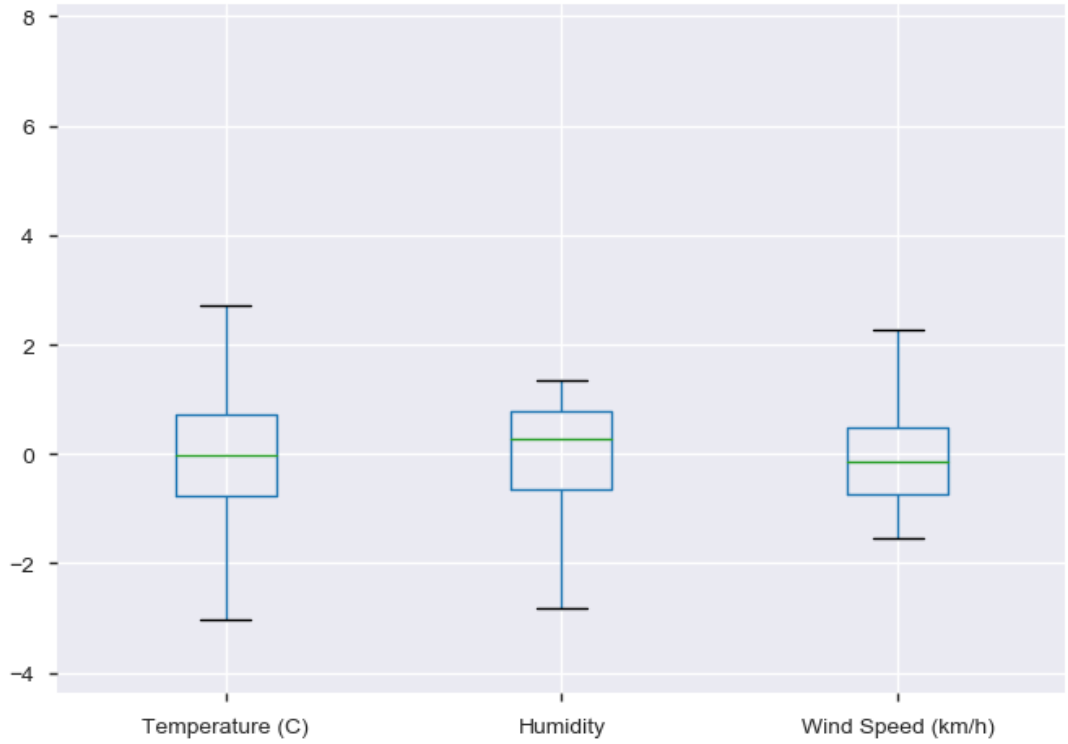
```
In [10]: scaler = StandardScaler().fit(data_raw)
         data_scaled = scaler.transform(data_raw)

In [11]: data_scaled_df = pd.DataFrame(data_scaled, columns=data_raw.columns)

In [12]: data_scaled_df.head()
```

Out[12]:

	Temperature (C)	Humidity	Wind Speed (km/h)
0	0.214510	0.681941	-1.081307
1	-0.468636	0.630496	0.058355
2	0.043288	-0.089742	0.775576
3	0.083917	-1.941780	0.694337
4	-0.205129	0.784832	-0.617086

```
In [13]: data_scaled_df.plot.box()
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd9bc1feb70>



2.5 Utilidades

```
In [14]: lista_metricas_treino = []
         lista_metricas_teste = []
```

```
In [15]: def metricas(y_true, y_pred, alg):
             r2 = r2_score(y_true, y_pred)
             eqm = mean_squared_error(y_true, y_pred)
             seq = len(y_true)*eqm
             reqm = math.sqrt(eqm)

             return {'Algoritmo':alg, 'R2':r2, 'EQM':eqm, 'REQM':reqm, 'SEQ':seq}
```

2.6 Separando conjuntos de Treino e Teste

Para a separação utilizou-se do train_test_split que divide o conjunto em treino e teste aleatóriamente

```
In [16]: train, test = train_test_split(data_scaled_df, test_size = 0.2, shuffle=True)

         x_train = train.drop(columns=['Temperature (C)'])
         y_train = train['Temperature (C)']
```

```
x_test = test.drop(columns=['Temperature (C)'])
y_test = test['Temperature (C)']
```

2.7 Aplicando a Regressão

2.7.1 Regressão Linear

```
In [17]: lire = LinearRegression()
```

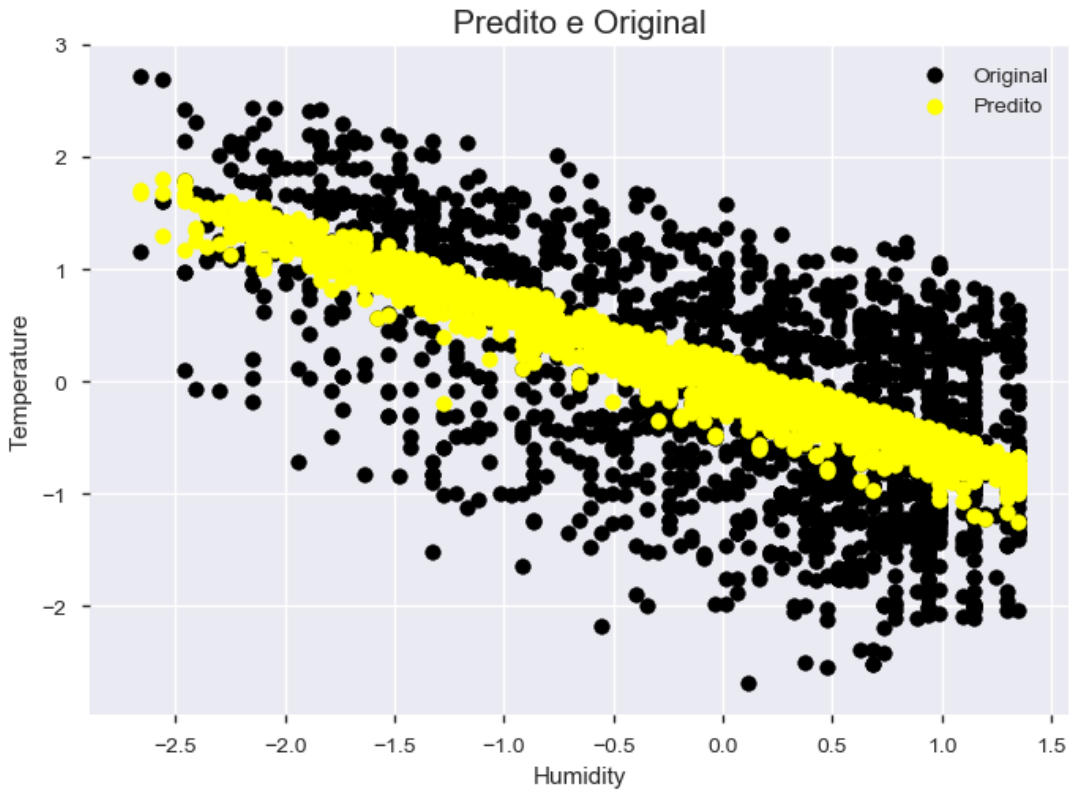
```
In [18]: lire.fit(x_train, y_train)
```

```
Out[18]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
                          normalize=False)
```

2.8 Avaliação para Teste

```
In [19]: y_pred = lire.predict(x_test)
         linear_metricas = metricas(y_test, y_pred, 'Regressão Linear - Teste')
         lista_metricas_teste.append(linear_metricas)
```

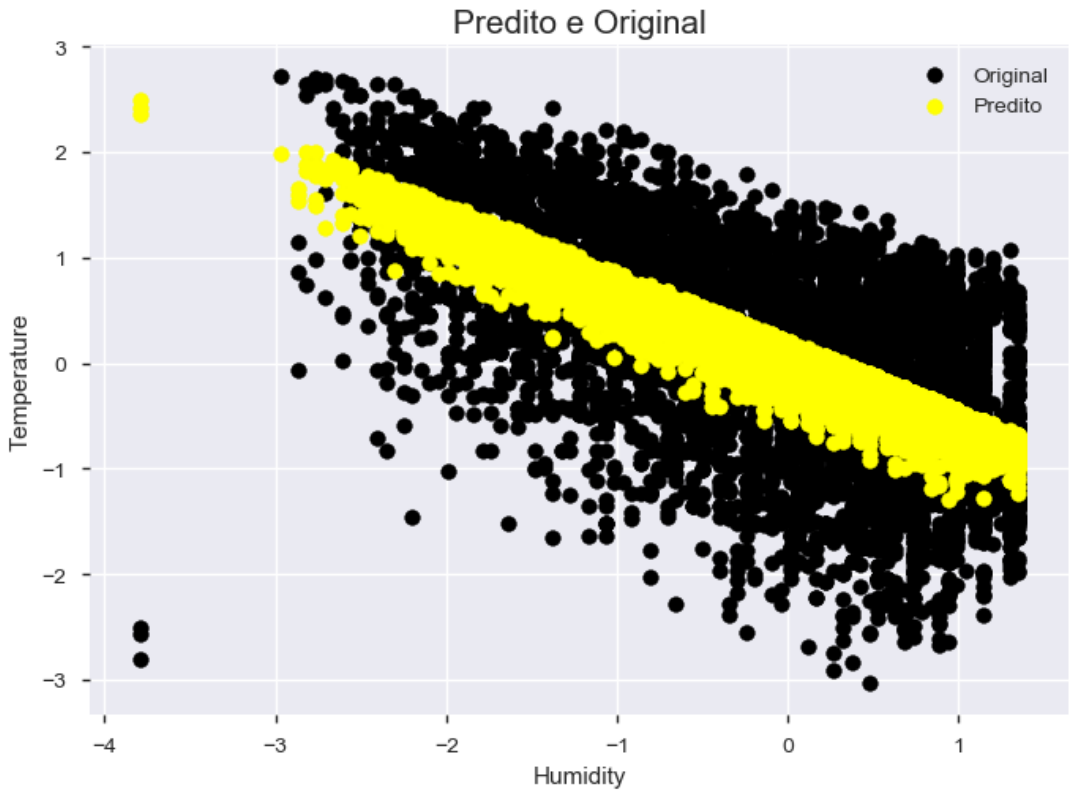
```
In [20]: plt.scatter(x_test['Humidity'], y_test,  color='black')
         plt.scatter(x_test['Humidity'], y_pred, color='yellow')
         plt.xlabel("Humidity")
         plt.ylabel("Temperature")
         plt.title('Predito e Original',fontsize=15)
         plt.legend(['Original', 'Predito'])
         plt.show()
```



2.9 Avaliação para Treino

```
In [21]: y_pred = lire.predict(x_train)
         linear_metricas = metricas(y_train, y_pred, 'Regressão Linear - Treino')
         lista_metricas_treino.append(linear_metricas)
```

```
In [22]: plt.scatter(x_train['Humidity'], y_train,  color='black')
         plt.scatter(x_train['Humidity'], y_pred, color='yellow')
         plt.xlabel("Humidity")
         plt.ylabel("Temperature")
         plt.title('Predito e Original',fontsize=15)
         plt.legend(['Original', 'Predito'])
         plt.show()
```



2.10 SVR

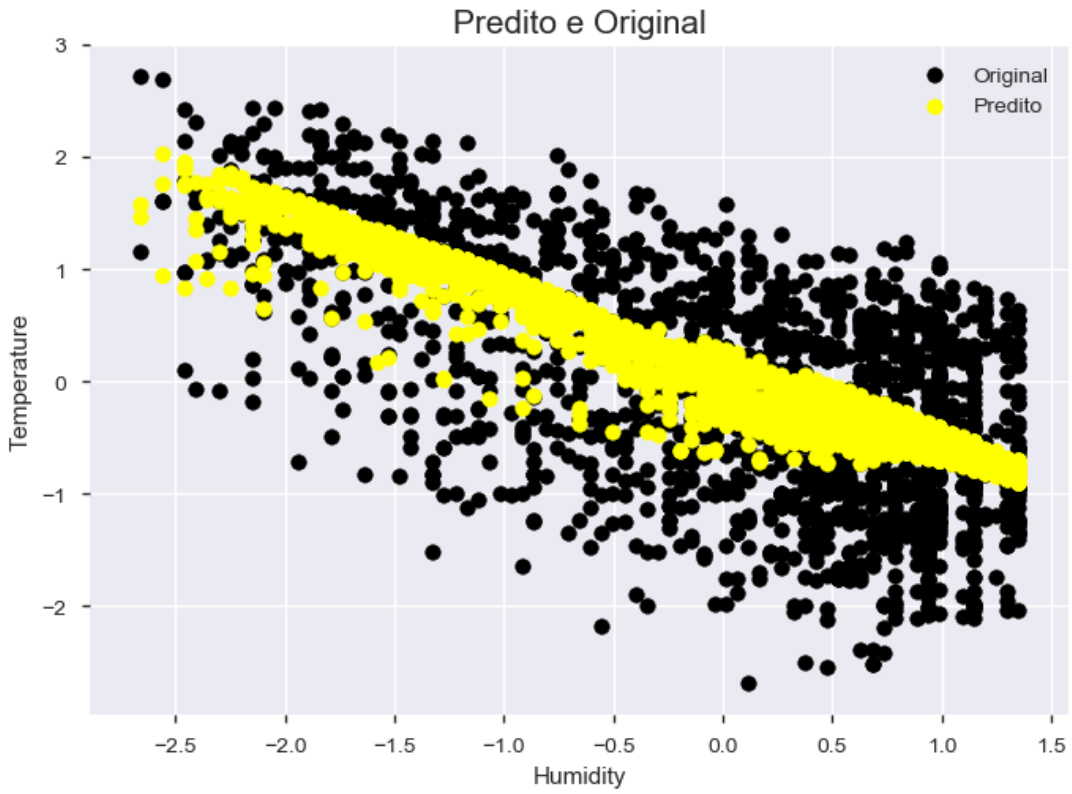
2.10.1 Kernel RBF

```
In [23]: svr_reg = SVR(kernel='rbf')

In [24]: svr_reg.fit(x_train, y_train)

Out[24]: SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
           gamma='auto_deprecated', kernel='rbf', max_iter=-1, shrinking=True,
           tol=0.001, verbose=False)
```

2.11 Avaliação para Teste

```
In [25]: y_pred = svr_reg.predict(x_test)
         svr_metricas = metricas(y_test, y_pred, 'SVR - RBF - Teste')
         lista_metricas_teste.append(svr_metricas)

In [26]: plt.scatter(x_test['Humidity'], y_test,  color='black')
         plt.scatter(x_test['Humidity'], y_pred, color='yellow')
         plt.xlabel("Humidity")
         plt.ylabel("Temperature")
```

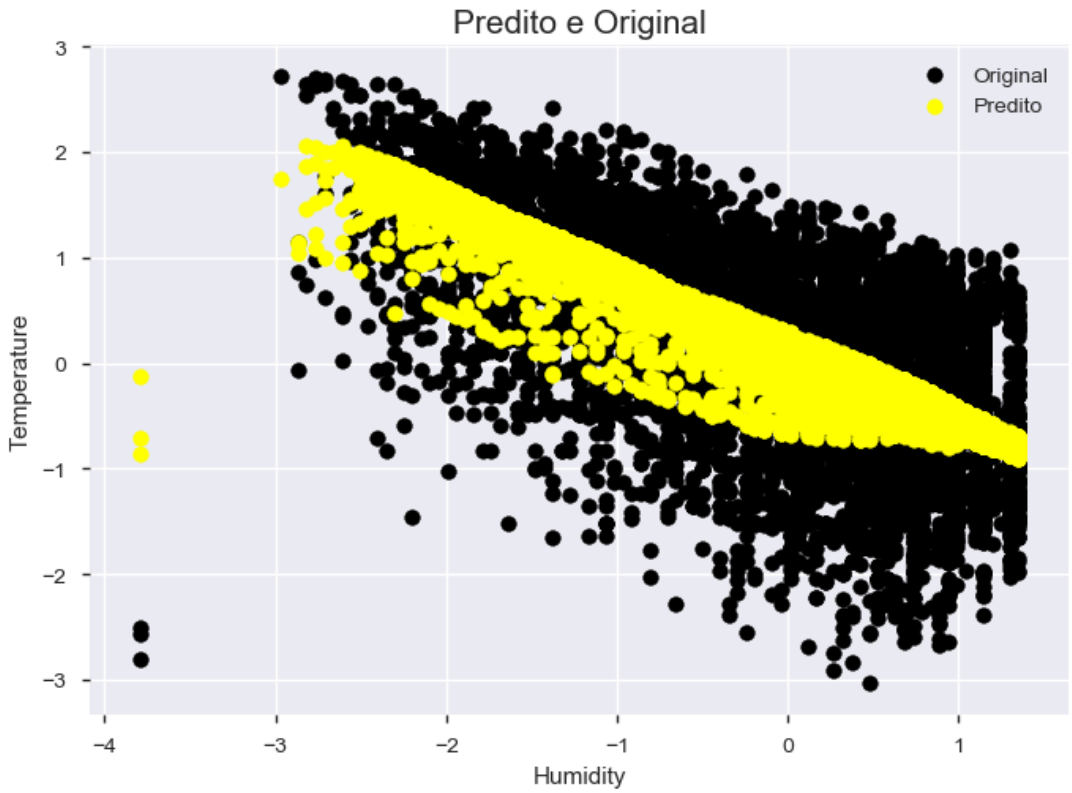
```
plt.title('Predito e Original',fontsize=15)
plt.legend(['Original', 'Predito'])
plt.show()
```



2.12 Avaliação para Treino

In [27]:
```
y_pred = svr_reg.predict(x_train)
svr_metricas = metricas(y_train, y_pred, 'SVR - RBF - Treino')
lista_metricas_treino.append(svr_metricas)
```

In [28]:
```
plt.scatter(x_train['Humidity'], y_train,  color='black')
plt.scatter(x_train['Humidity'], y_pred, color='yellow')
plt.xlabel("Humidity")
plt.ylabel("Temperature")
plt.title('Predito e Original',fontsize=15)
plt.legend(['Original', 'Predito'])
plt.show()
```



2.12.1 Kernel Linear

```
In [29]: svr_reg = SVR(kernel='linear')
```

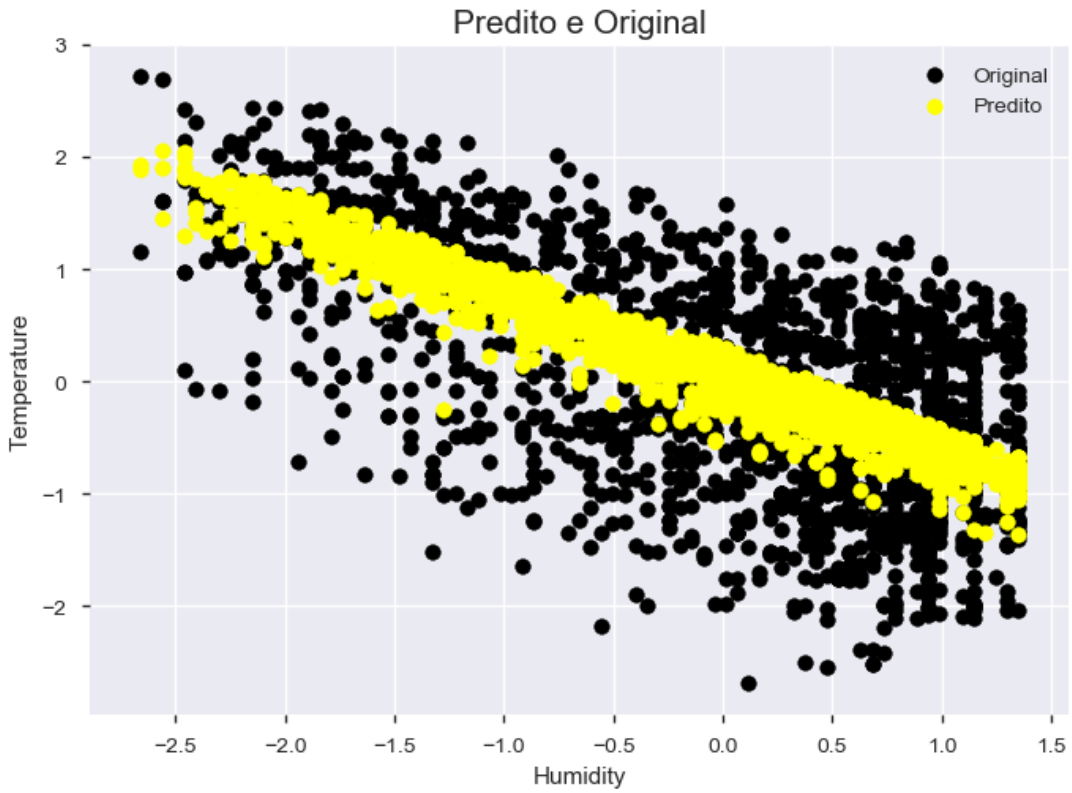
```
In [30]: svr_reg.fit(x_train, y_train)
```

```
Out[30]: SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
           gamma='auto_deprecated', kernel='linear', max_iter=-1, shrinking=True,
           tol=0.001, verbose=False)
```

2.13 Avaliação para Teste

```
In [31]: y_pred = svr_reg.predict(x_test)
         metricas_svr = metricas(y_test, y_pred, 'SVR - Linear - Teste')
         lista_metricas_teste.append(metricas_svr)
```

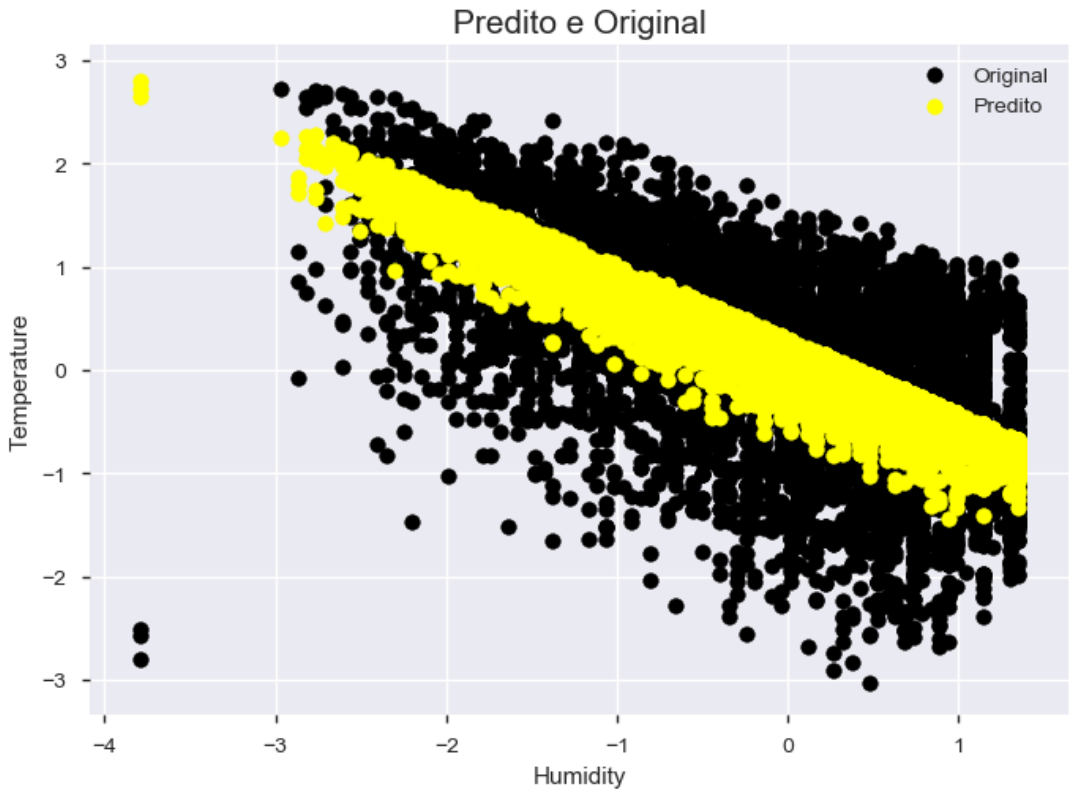
```
In [32]: plt.scatter(x_test['Humidity'], y_test,  color='black')
         plt.scatter(x_test['Humidity'], y_pred, color='yellow')
         plt.xlabel("Humidity")
         plt.ylabel("Temperature")
         plt.title('Predito e Original',fontsize=15)
         plt.legend(['Original', 'Predito'])
         plt.show()
```



2.14 Avaliação para Treino

```
In [33]: y_pred = svr_reg.predict(x_train)
         svr_metricas = metricas(y_train, y_pred, 'SVR - Linear - Treino')
         lista_metricas_treino.append(svr_metricas)
```

```
In [34]: plt.scatter(x_train['Humidity'], y_train,  color='black')
         plt.scatter(x_train['Humidity'], y_pred, color='yellow')
         plt.xlabel("Humidity")
         plt.ylabel("Temperature")
         plt.title('Predito e Original',fontsize=15)
         plt.legend(['Original', 'Predito'])
         plt.show()
```



2.14.1 Kernel Sigmoide

```
In [35]: svr_reg = SVR(kernel='sigmoid')
```

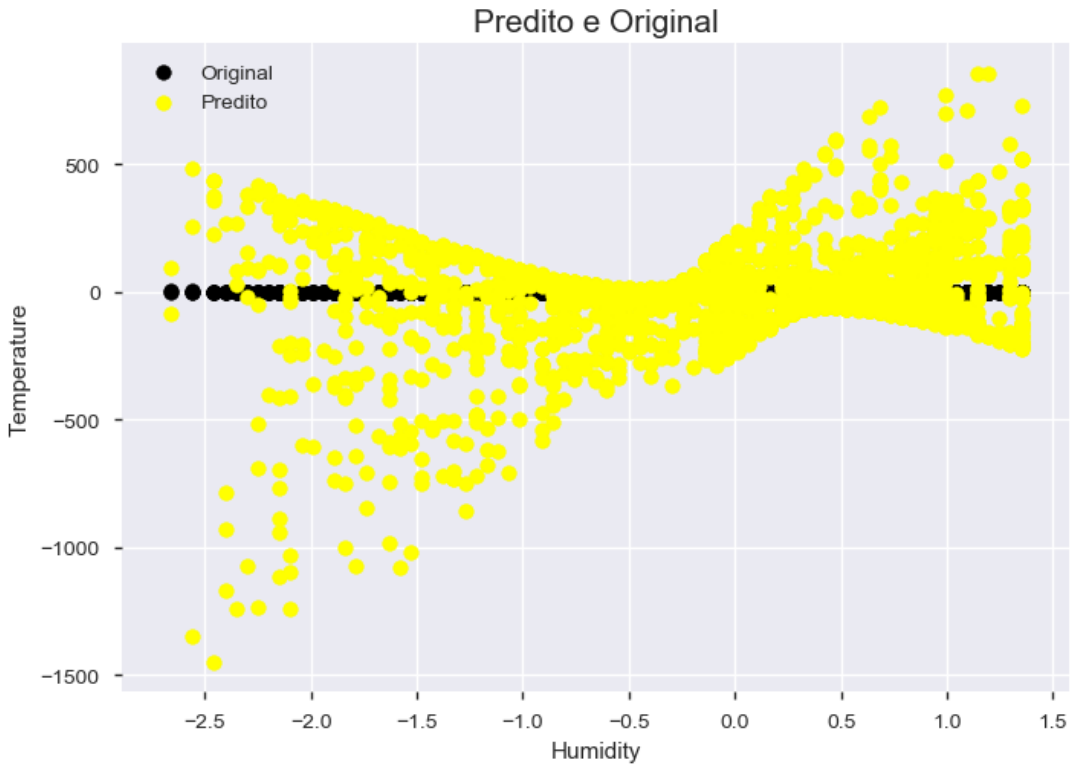
```
In [36]: svr_reg.fit(x_train , y_train)
```

```
Out[36]: SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
           gamma='auto_deprecated', kernel='sigmoid', max_iter=-1, shrinking=True,
           tol=0.001, verbose=False)
```

2.15 Avaliação para Teste

```
In [37]: y_pred  = svr_reg.predict(x_test)
         metricas_svr = metricas(y_test , y_pred , 'SVR - Sigmoide - Teste')
         lista_metricas_teste.append(metricas_svr)
```

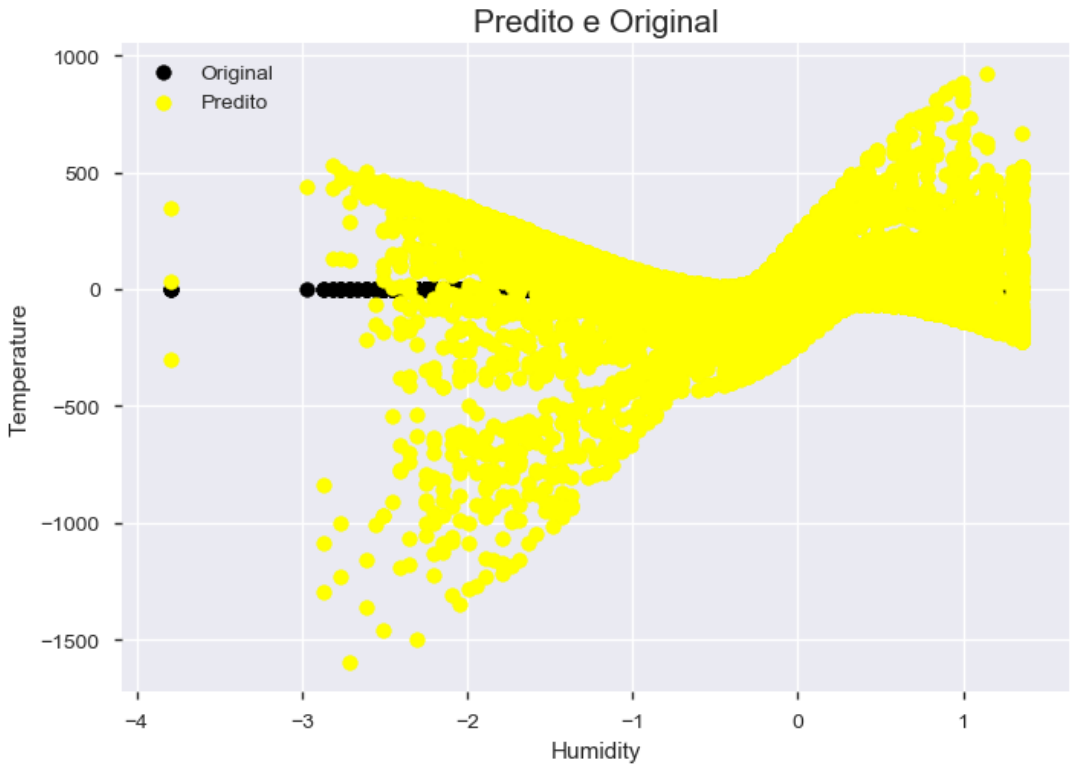
```
In [38]: plt.scatter(x_test['Humidity'], y_test,  color='black')
         plt.scatter(x_test['Humidity'], y_pred, color='yellow')
         plt.xlabel("Humidity")
         plt.ylabel("Temperature")
         plt.title('Predito e Original',fontsize=15)
         plt.legend(['Original', 'Predito'])
         plt.show()
```



2.16 Avaliação para Treino

```
In [39]: y_pred = svr_reg.predict(x_train)
         svr_metricas = metricas(y_train , y_pred , 'SVR - Sigmoide - Treino')
         lista_metricas_treino.append(svr_metricas)
```

```
In [40]: plt.scatter(x_train['Humidity'], y_train,  color='black')
         plt.scatter(x_train['Humidity'], y_pred, color='yellow')
         plt.xlabel("Humidity")
         plt.ylabel("Temperature")
         plt.title('Predito e Original',fontsize=15)
         plt.legend(['Original', 'Predito'])
         plt.show()
```

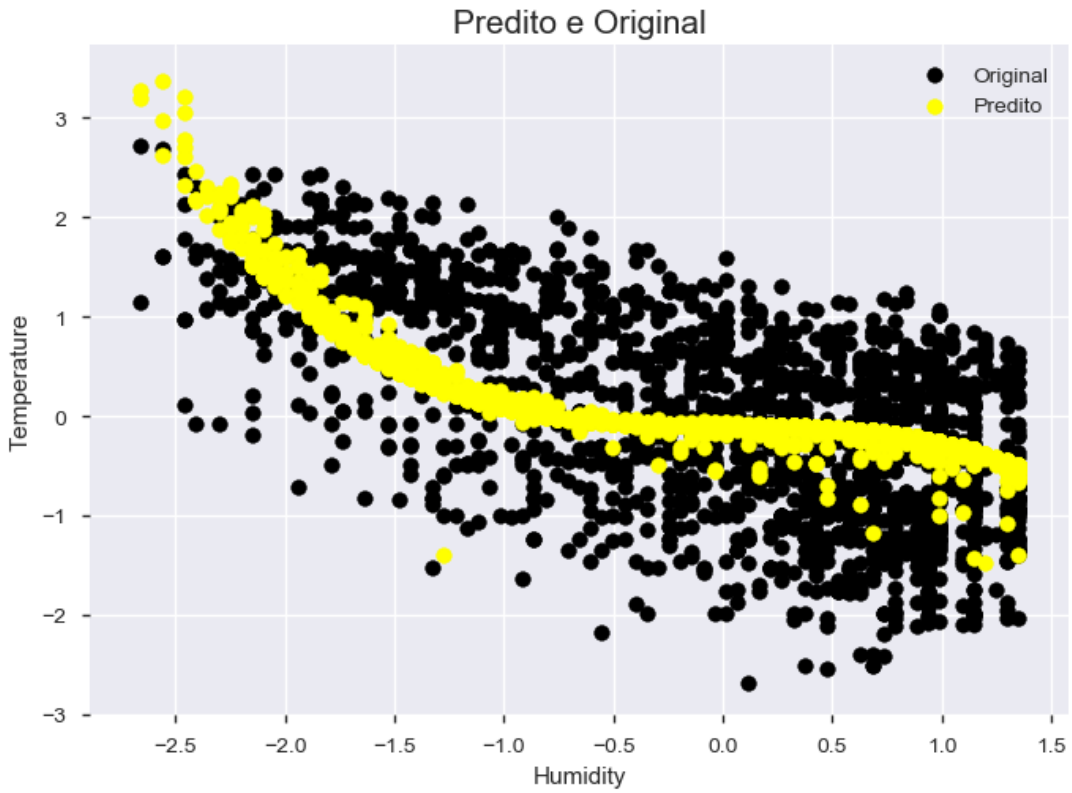


2.16.1 Kernel Polinomial

```
In [41]: svr_reg = SVR(kernel='poly', degree=3)

In [42]: svr_reg.fit(x_train, y_train)

Out[42]: SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
           gamma='auto_deprecated', kernel='poly', max_iter=-1, shrinking=True,
           tol=0.001, verbose=False)
```

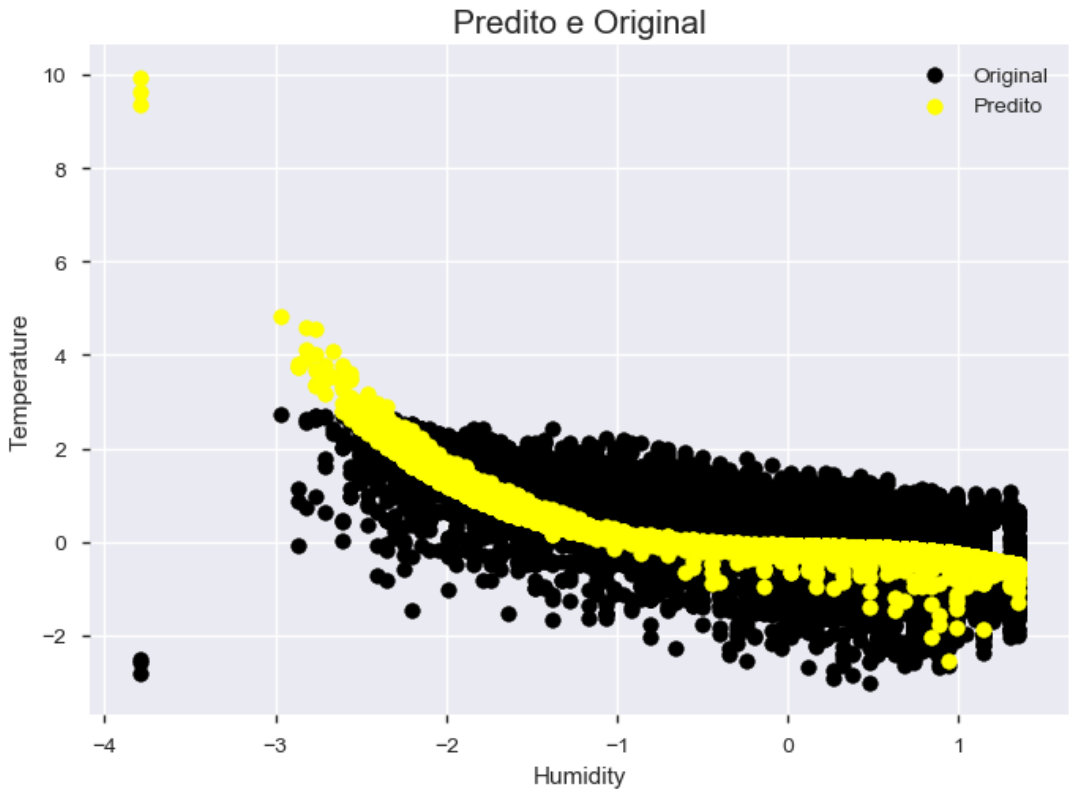
2.17 Avaliação para Teste

```
In [43]: y_pred = svr_reg.predict(x_test)
         svr_metricas = metricas(y_test, y_pred, 'SVR - Polinomial - Teste')
         lista_metricas_teste.append(svr_metricas)

In [44]: plt.scatter(x_test['Humidity'], y_test,  color='black')
         plt.scatter(x_test['Humidity'], y_pred, color='yellow')
         plt.xlabel("Humidity")
         plt.ylabel("Temperature")
         plt.title('Predito e Original',fontsize=15)
         plt.legend(['Original', 'Predito'])
         plt.show()
```



2.18 Avaliação para Treino

```
In [45]: y_pred = svr_reg.predict(x_train)
         svr_metricas = metricas(y_train, y_pred, 'SVR - Polinomial - Treino')
         lista_metricas_treino.append(svr_metricas)

In [46]: plt.scatter(x_train['Humidity'], y_train,  color='black')
         plt.scatter(x_train['Humidity'], y_pred, color='yellow')
         plt.xlabel("Humidity")
         plt.ylabel("Temperature")
         plt.title('Predito e Original',fontsize=15)
         plt.legend(['Original', 'Predito'])
         plt.show()
```



2.19 Redes Neurais

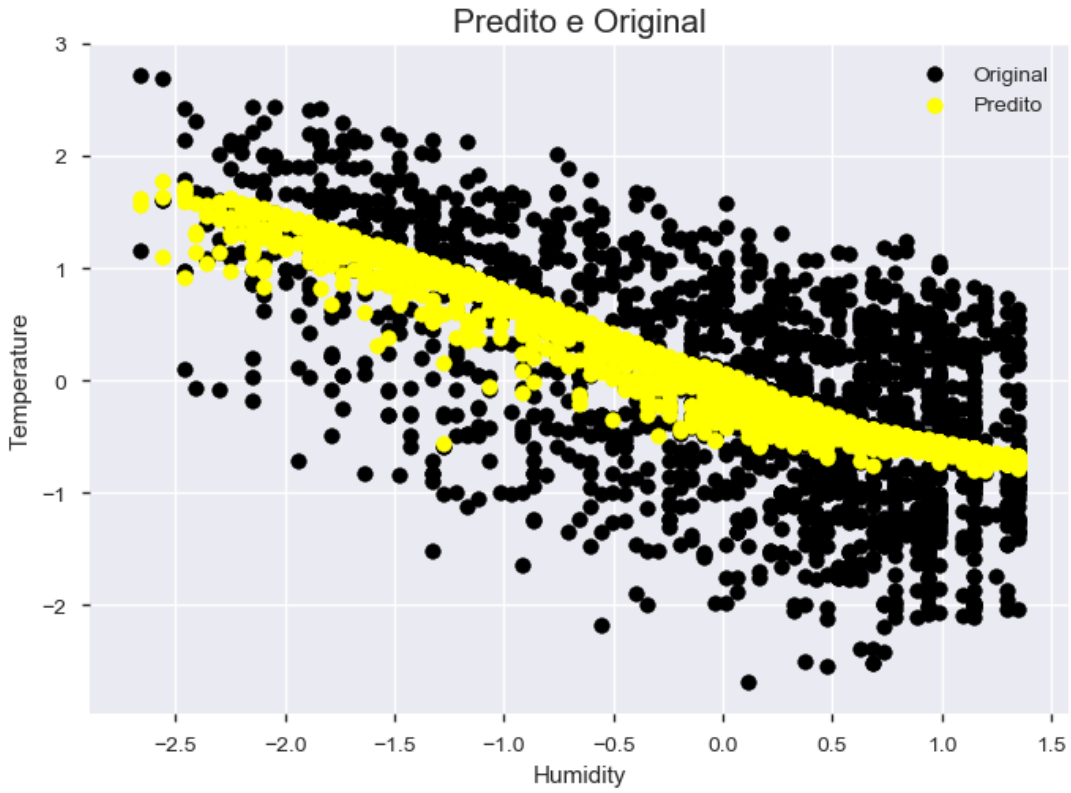
2.19.1 Kernel Linear

```
In [47]: mlp_reg = MLPRegressor()

In [48]: mlp_reg.fit(x_train, y_train)

Out[48]: MLPRegressor(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
                      beta_2=0.999, early_stopping=False, epsilon=1e-08,
                      hidden_layer_sizes=(100,), learning_rate='constant',
                      learning_rate_init=0.001, max_iter=200, momentum=0.9,
                      n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
                      random_state=None, shuffle=True, solver='adam', tol=0.0001,
                      validation_fraction=0.1, verbose=False, warm_start=False)
```

2.20 Avaliação para Teste

```
In [49]: y_pred = mlp_reg.predict(x_test)
         mlp_metricas = metricas(y_test, y_pred, 'MLP - Teste')
         lista_metricas_teste.append(mlp_metricas)
```

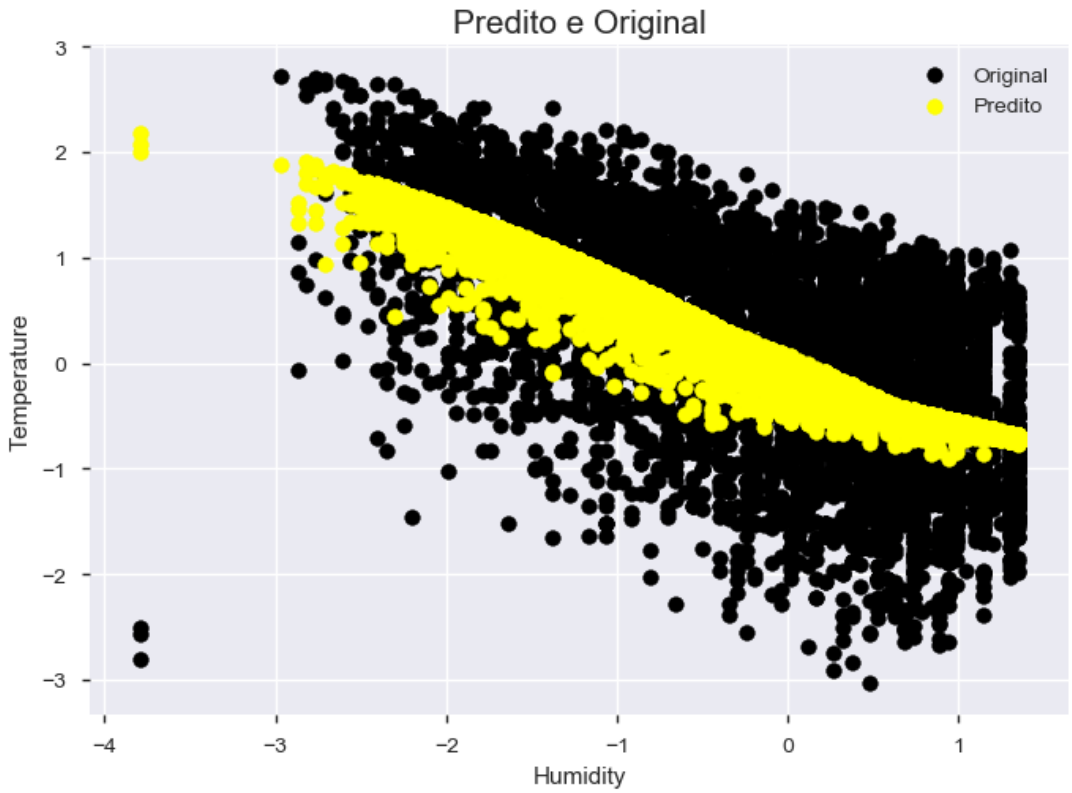
```
In [50]: plt.scatter(x_test['Humidity'], y_test,  color='black')
         plt.scatter(x_test['Humidity'], y_pred, color='yellow')
         plt.xlabel("Humidity")
         plt.ylabel("Temperature")
         plt.title('Predito e Original',fontsize=15)
         plt.legend(['Original', 'Predito'])
         plt.show()
```



2.21 Avaliação para Treino

```
In [51]: y_pred = mlp_reg.predict(x_train)
         mlp_metricas = metricas(y_train, y_pred, 'MLP - Treino')
         lista_metricas_treino.append(mlp_metricas)
```

```
In [52]: plt.scatter(x_train['Humidity'], y_train,  color='black')
         plt.scatter(x_train['Humidity'], y_pred, color='yellow')
         plt.xlabel("Humidity")
         plt.ylabel("Temperature")
         plt.title('Predito e Original',fontsize=15)
         plt.legend(['Original', 'Predito'])
         plt.show()
```



3 Resultados

```
In [63]: metricas_teste = pd.DataFrame(lista_metricas_teste)
         metricas_teste
```

Out[63]:

	Algoritmo	EQM	R2	REQM	SEQ
0	Regressão Linear - Teste	0.592699	0.405625	0.769870	1.185398e+03
1	SVR - RBF - Teste	0.599608	0.398697	0.774343	1.199215e+03
2	SVR - Linear - Teste	0.601256	0.397044	0.775407	1.202513e+03
3	SVR - Sigmoide - Teste	51226.694361	-51370.514410	226.333149	1.024534e+08
4	SVR - Polinomial - Teste	0.697636	0.300392	0.835246	1.395272e+03
5	MLP - Teste	0.583083	0.415269	0.763599	1.166166e+03

```
In [65]: metricas_teste = round(metricas_teste, 3)

In [66]: metricas_teste
```

Out[66]:

	Algoritmo	EQM	R2	REQM	SEQ
0	Regressão Linear - Teste	0.593	0.406	0.770	1.185398e+03
1	SVR - RBF - Teste	0.600	0.399	0.774	1.199215e+03
2	SVR - Linear - Teste	0.601	0.397	0.775	1.202513e+03
3	SVR - Sigmoide - Teste	51226.694	-51370.514	226.333	1.024534e+08
4	SVR - Polinomial - Teste	0.698	0.300	0.835	1.395272e+03
5	MLP - Teste	0.583	0.415	0.764	1.166166e+03

```
In [67]: metricas_teste.to_excel('regressao2_metricas_teste.xlsx')

In [68]: metricas_treino = pd.DataFrame(lista_metricas_treino)
         metricas_treino
```

Out[68]:

	Algoritmo	EQM	R2	REQM	SEQ
0	Regressão Linear - Treino	0.584740	0.415589	0.764683	4.677922e+03
1	SVR - RBF - Treino	0.577293	0.423032	0.759798	4.618342e+03
2	SVR - Linear - Treino	0.592106	0.408227	0.769484	4.736851e+03
3	SVR - Sigmoide - Treino	52816.201340	-52785.446344	229.817757	4.225296e+08
4	SVR - Polinomial - Treino	0.755123	0.245303	0.868978	6.040982e+03
5	MLP - Treino	0.574974	0.425350	0.758270	4.599792e+03

```
In [69]: metricas_treino = round(metricas_treino, 3)

In [70]: metricas_treino.to_excel('regressao2_metricas_treino.xlsx')
```